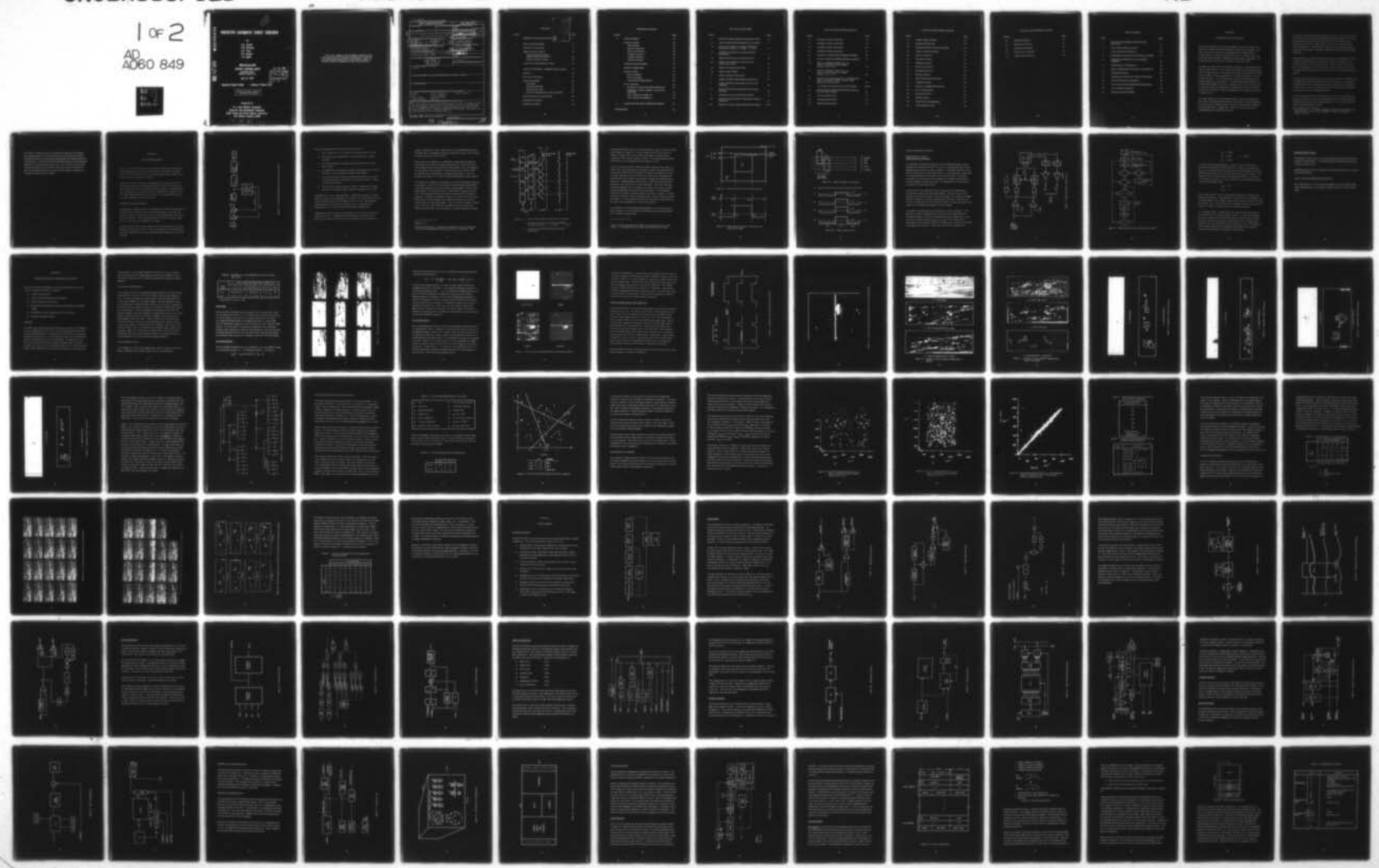
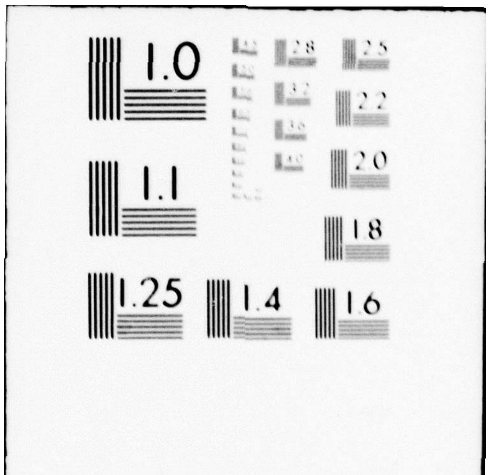




MICROCOPY RESOLUTION TEST CHART

2
B.S.

PROTOTYPE AUTOMATIC TARGET SCREENER

By

D.E. Soland
P.M. Narendra
R.C. Fitch
D.V. Serreyn
T.G. Kopet

Honeywell
SYSTEMS & RESEARCH CENTER
2600 RIDGWAY PARKWAY
MINNEAPOLIS, MINNESOTA 55413

June 15, 1978

Quarterly Progress Report 1 January–31 March 1978

Prepared for

U.S. Army Mobility Equipment
Research and Development Command,
Night Vision and Electro-Optics Laboratory
Fort Belvoir, Virginia 22060

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (WHEN DATA ENTERED)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOV'T ACCESSION NUMBER	3. RECIPIENT'S CATALOG NUMBER (9)
4. TITLE (AND SUBTITLE) (6) PROTOTYPE AUTOMATIC TARGET SCREENER.		5. TYPE OF REPORT/PERIOD COVERED Quarterly Progress Report. no. 2 1 Jan[redacted] ~~to~~ 31 Mar[redacted] 1978,
		6. PERFORMING ORG. REPORT NUMBER (14) 78SRC54-2
7. AUTHOR(S) (10) D. E. Soland, P. M. Narendra, R. C. Fitch, D. V. Serreyn, T. G. Kopet		8. CONTRACT OR GRANT NUMBER(S) (15) DAAK70-77-C-0248
9. PERFORMING ORGANIZATIONS NAME/ADDRESS Honeywell Systems and Research Center 2600 Ridgway Parkway Minneapolis, Minnesota 55413		10. PROGRAM ELEMENT PROJECT, TASK AREA & WORK UNIT NUMBERS (16) 1E263710DK70 (17) 14 010CJ
11. CONTROLLING OFFICE NAME/ADDRESS Night Vision and Electro-Optics Laboratory Fort Belvoir, Virginia 22060 (11) 15		12. REPORT DATE June [redacted] 1978
		13. NUMBER OF PAGES 108
14. MONITORING AGENCY NAME/ADDRESS (IF DIFFERENT FROM CONT. OFF.) (12) 109p.		15. SECURITY CLASSIFICATION (OF THIS REPORT) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (OF THIS REPORT)

Unlimited

17. DISTRIBUTION STATEMENT (OF THE ABSTRACT ENTERED IN BLOCK 20, IF DIFFERENT FROM REPORT)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (CONTINUE ON REVERSE SIDE IF NECESSARY AND IDENTIFY BY BLOCK NUMBER)

Infrared	Target recognition	Image enhancement
FLIR	Pattern recognition	
Target cueing	Image processing	
Target screening	Real-time	

20. ABSTRACT (CONTINUE ON REVERSE SIDE IF NECESSARY AND IDENTIFY BY BLOCK NUMBER)

This report is the second quarterly progress report for contract DAAK70-77-C-0248, Prototype Automatic Target Screener. It describes results of a five-month design study. The objective of the effort is an automatic target screener to be used with thermal imaging systems employing common module components. This report covers the period from 1 January to 31 March 1978.

HO-168 REV 11/74

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 55 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (WHEN DATA ENTERED)

78 10 30 129

402 349

1B

CONTENTS

Section Page

1 INTRODUCTION AND SUMMARY 1

2 IMAGE ENHANCEMENT 4

Synthetic DC Restoration 4

Global Gain/Bias Control 12

Global Gain/Bias Control Implementation and Software 12

Global Gain/Bias Testing 16

Local Area Gain/Brightness Control 16

3 TARGET SCREENING ALGORITHM SIMULATION 17

Summary 17

Data Base Preparation 18

Image Segmentation 18

Thresholds 19

Threshold Selection 19

Interval Generation 21

Results of Segmentation of NVL Data Base 23

Feature Extraction and Evaluation 34

Recognition Classifier 37

Interframe Analysis 43

CONTENTS (concluded)

Section		Page
4	SYSTEM DESIGN	50
	Hardware Design	50
	Autothreshold	52
	Interval Generation	60
	First Level Features	64
	Computer System 1	66
	Computer System 2	71
	Symbol Generation	71
	Timing and Synchronization	75
	Physical Configuration	75
	Software Design	79
	Data Acquisition	79
	Data Processing	81
	Computational Requirements	90
	CPU 2 Functions	94
	Interframe Analysis and Symbol Generation	95
	Diagnostics, Feature Dumps, and Test Data Generation	96
	CPU 1 Software Development	97
	CPU 2 Software Development	97
5	PLANS FOR THE NEXT REPORTING PERIOD	98
REFERENCES		99

LIST OF ILLUSTRATIONS

Figure		Page
1	Functional Image Enhancement System on PATS	5
2a	Light-Emitting Diode Overlap with 2:1 Interlace	8
2b	Normalized Luminance Intensity Distribution for Separate Fields (——— Odd, ----- Even)	8
2c	Cumulative Normalized Luminous Intensity Distribution	8
3a	Sample Thermal Scene and Detector Scan	10
3b	Single Detector Output to LED Driver for Consecutive Fields	10
4a	Vidicon Scanning of LED Sweep	11
4b	Vidicon Video Format	11
5	Global Gain/Bias Control Unit	13
6	Global Gain/Bias Control Software Flowchart	14
7	Sample Digitized FLIR Imagery from the NVL Videotape	20
8	Result of One-Directional Interval Generation Criterion	22
9	Symmetric Interval Generation Criterion	24
10	Result of Using Symmetric Criterion for Interval Generation	25
11	Results of a Scene Adaptive Segmentation Scheme	26-27

LIST OF ILLUSTRATIONS (continued)

Figure		Page
12	Examples of Object Extraction	28
13	Examples of Object Extraction	29
14	Examples of Object Extraction	30
15	Examples of Object Extraction	31
16	Ground Truth Distribution of Segmented Objects	33
17	PATS Cascaded Thresholded Classifier Approach	36
18	Fourier Amplitude Features (a_3, a_4) (Show no separation between target classes A, B, C, D)	39
19	Fourier Amplitude Features (a_4, a_5) (Show no separation between target classes A, B, C, D)	40
20	Scatter Plot of $\mu 20$ (intensity) vs. $\mu 20$ (silhouette) (Shows correlation of "like" moments from different moment classes)	41
21	An Example of a Sequence of Tactical Imagery	45-46
22	Result of Segmentation of FLIR Frame Sequence for Interframe Analysis	47
23	PATS Functional Units	51
24	Autothreshold Structure	53
25	Edge Signal Derivation	54

LIST OF ILLUSTRATIONS (continued)

Figure		Page
26	Adaptive Edge Threshold	55
27	Background Estimator	57
28	Background Filter and Switch Rationale	58
29	Adaptive Bright Threshold	59
30	Interval Generation	61
31	Start/Stop Criteria	62
32	Validation Criteria	63
33	First Level Features	65
34	Background Feature	67
35	Intensity Feature	68
36	Video Field Memory Bit Plane	69
37	Computer System 1	70
38	Memory 1 Configuration/Interface	72
39	CPU 2 Configuration	73
40	Symbol Generation	74
41	Timing and Sync	76
42	PATS Physical Configuration	77
43	PATS Physical Layout	78

LIST OF ILLUSTRATIONS (concluded)

Figure		Page
44	Software Functions	80
45	Interval Data Format	82
46	Bin Matching Software	83
47	Map of Interval Memory	85
48	Median Filter Software	88

LIST OF TABLES

Table		Page
1	Summary of the Digitized FLIR Data Base (NVL Data)	19
2	Clutter Discrimination Features	35
3	Clutter Rejection Performance	35
4	Recognition Features Used in Prototype Classifier	42
5	Target Class Definition for the Recognition Classifier	42
6	Performance on Training Data	44
7	Object Classification for Sequence from Figure 22	48
8	Possible Bin Format	86
9	Preliminary Features and Clutter Prescreening	89
10	Primary Features and Classifier	90
11	CPU 1 Worst Case Computational Requirements	91
12	CPU 2 Software Functions	94
13	Symbol Generation Software	95

SECTION 1

INTRODUCTION AND SUMMARY

This is the second quarterly technical progress report for Contract No. DAAK70-77-C-0248, Prototype Automatic Target Screener (PATS). The report describes results of the second half of a five-month Phase I design study for an automatic target screener that can operate with first generation thermal imagers employing common module components. The period covered by this report is 1 January to 31 March 1978.

The objective of this effort is to produce a design for a prototype automatic target screener (PATS). The screener will reduce the task loading on the thermal imager operator by detecting and recognizing a limited set of high priority targets at ranges comparable to or greater than those for an unassisted observer. A second objective is to provide enhancement of the video presentation to the operator. The image enhancement includes: 1) automatic gain/brightness control, to relieve the operator of the necessity to continually adjust the display gain and brightness controls; and 2) DC restoration, to eliminate artifacts resulting from AC coupling of the infrared (IR) detectors.

The report consists of four principal sections. Section 2 describes the effort under the image enhancement part of the study; Section 3 describes the target screener design activities. Section 4 describes the results of the preliminary system design task. Plans for the next three-month reporting period are included in Section 5.

The image enhancement portion of PATS will consist of circuitry to operate on the Common Module FLIR (MODFLIR) video output signal. This circuitry will provide global gain and bias control in the form of feedback to the MODFLIR to maintain the signal within the dynamic range of the electro-optical multiplexer. The global gain and bias control circuit preliminary design has been completed and will be implemented upon receipt of the GFE MODFLIR.

Image enhancement will also include local area gain and brightness control to enhance local variations of contrast and compress the overall scene dynamic range to match that of the display. This circuitry has been completed and examples of its performance on videotaped thermal image data were included, along with the circuit description, in the previous quarterly report.[1]

The third image enhancement circuit is for DC restoration, to eliminate the streaking associated with loss of line-to-line correlation on the displayed image because of the AC coupling of the detector channels. A breadboard version of the DC restore concept previously described has been designed and constructed and is undergoing checkout.

Sections 3 and 4 summarize the results of the target screener design task. Section 3 describes the results of the subtasks involved with the design of algorithms for detecting and recognizing targets. Section 4 describes the system design preliminary results.

---

[1] D. E. Soland, et al., "Prototype Automatic Target Screener," Quarterly Progress Report, Contract No. DAAK70-77-C-0248, ADA 050684, January 15, 1978.

The target screening activities including data preparation and analysis, image segmentation, and feature extraction were reported previously.[1] The object classification and target decision tasks remaining are describ-ed in Section 3. A total of 385 frames of FLIR imagery has been digitized, annotated, and debanded where necessary. These images contain tanks, armored personnel carriers (APCs), and some trucks, and represent most of the imagery required for target screener design, training, and testing for the tank and APC classes.

SECTION 2

IMAGE ENHANCEMENT

This section reports the progress on the image enhancement tasks of the Prototype Automatic Target Screener (PATS). Specifically, the tasks addressed are synthetic DC restoration, global gain and bias control, and local area gain and brightness control.

Figure 1 is a functional diagram showing these three functions in PATS. The hardware for the all-analog synthetic DC restoration has been breadboarded and is being tested; progress will be presented. The global gain/bias control hardware has been built and one algorithm is being tested on the FLIR; this algorithm will be discussed. New developments on local area gain/brightness control will be given.

SYNTHETIC DC RESTORATION

The algorithm, implementation, and justification for the all-analog approach to synthetic DC restoration were discussed previously[1]. Since then, the hardware has been breadboarded and is currently being tested on the simulation test patterns. The results from these test patterns have been good thus far. Testing on the MODFLIR is starting.

An earlier report[1] did not cover fully all of the assumptions for the analog synthetic DC restoration scheme. These assumptions are discussed below, and those related to specific common module FLIR properties are covered in detail.

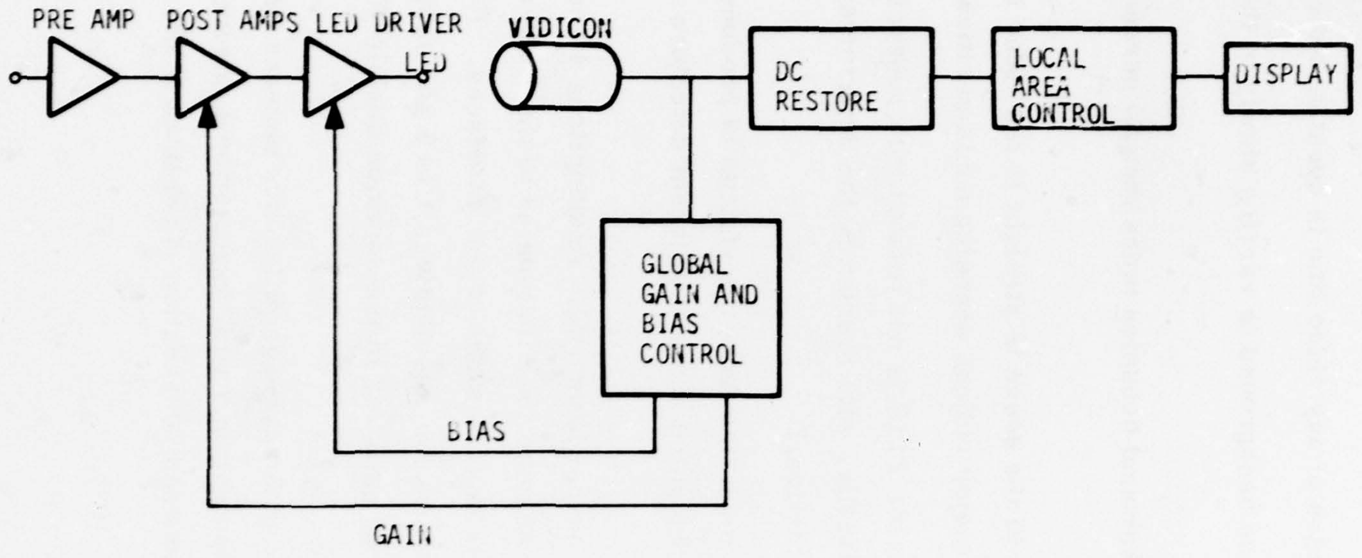


Figure 1. Functional Image Enhancement System on PATS

The major assumptions for synthetic DC restoration are:

1. The average value of any video line is constant over the scene.
2. The actual scene background is varying slowly in the vertical direction.
3. The scene background occupies more than 50 percent of a scan line of video.
4. A straight line in the scene is straight in the video from the FLIR. (There are no major vidicon scanning nonlinearities.)
5. The video from the FLIR is not rotated with respect to the scene, or more specifically, with respect to the detector array. (There is no vidicon rotation.)
6. Any horizontal scan line from the vidicon is produced by a single detector or by weighted sums of adjacent detectors in the FLIR.

Assumptions 1 to 3 are scene-dependent. Assumptions 4 and 5 are properties of the vidicon, and Assumption 6 is related to the LED-vidicon interface and the scanning systems on both sides of the interfaces. The following discussion gives the reasons for Assumptions 1 to 3 and shows that Assumptions 4 to 6 can be satisfied in the common module FLIR.

Assumption 1 ensures that the average difference between two consecutive video lines is zero. Assumption 2 with Assumption 1 ensures that the difference is constant between consecutive video lines in the background regions of those lines. With Assumption 3, thresholding will provide the DC shift value of the difference between background regions; this technique was discussed in an earlier report.[2]

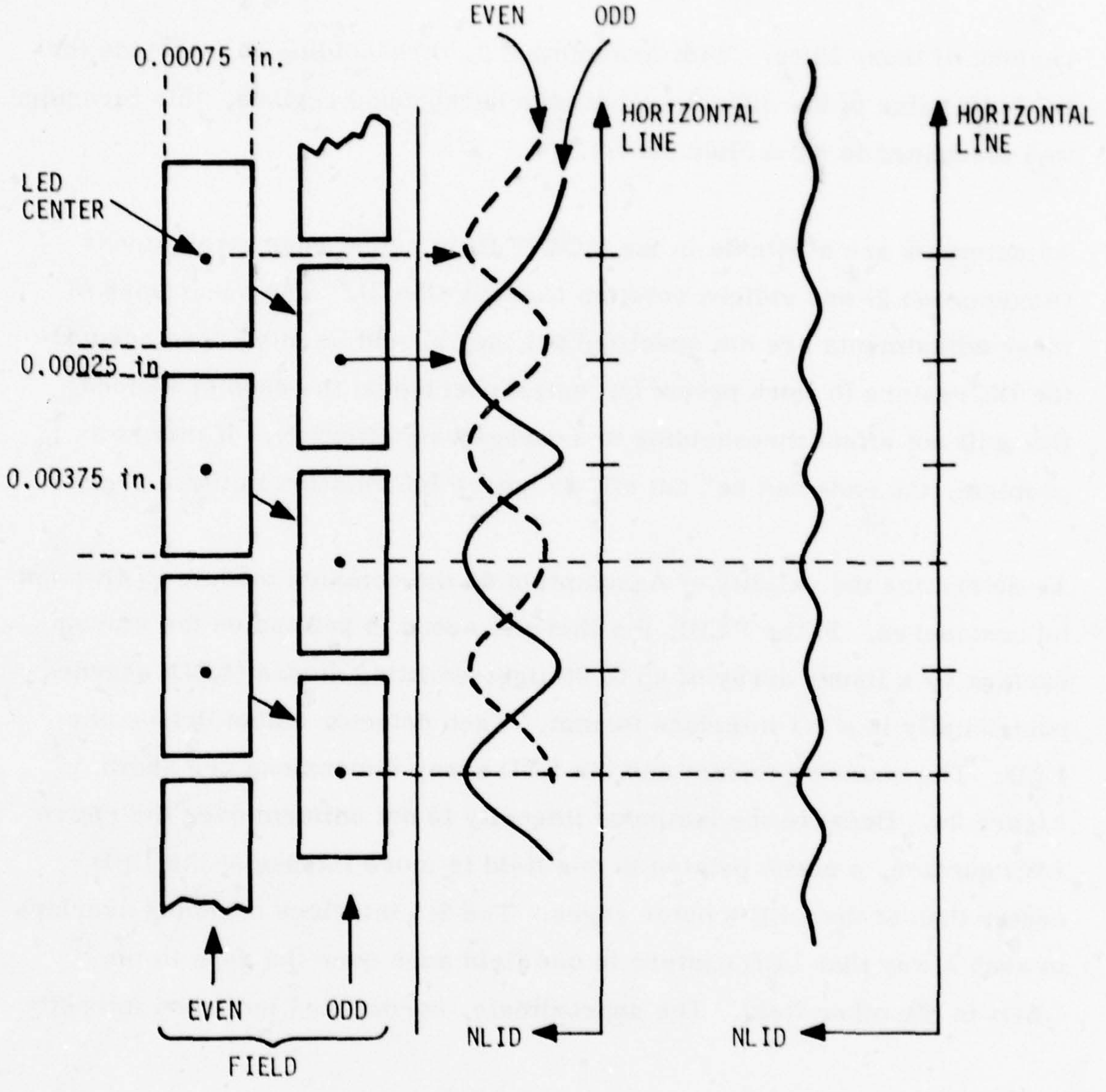
Adjustments are available in the MODFLIR to adjust edge straightness (Assumption 2) and vidicon rotation (Assumption 3).[3] The accuracies of these adjustments are not specified but they should be more than adequate for DC restore to work properly; small overlap on the ends of a video line will not affect thresholding or averages significantly. If this is a problem, the ends can be "cut off" to ignore information in these regions.

To determine the validity of Assumption 6, the common module FLIR must be considered. In the FLIR, the thermal scene is painted on the vidicon surface by a linear array of up to 80 light-emitting diodes (LED) scanned horizontally in a 2:1 interlace format. Each detector output drives one LED. The scanning format and the LED array dimensions are shown in Figure 2a. Because the luminous intensity is not uniform over the entire LED surface, a scene painted in one field is more intense at the LED center than at the LED's outer edges. The 2:1 interlace scanning overlaps in such a way that LED centers in one field scan over the gaps in the LEDs in the other field. The approximate, normalized luminous intensity

---

[2] Soland, p. 17.

[3] Texas Instruments Inc., "Operation and Maintenance Instructions with Parts List for Multipurpose Observation Device," September, 1976.



a b c

Figure 2. a. Light-Emitting Diode Overlap with 2:1 Interlace

b. Normalized Luminance Intensity Distribution for Separate Fields (________ Odd, - - - - - Even)

c. Cumulative Normalized Luminous Intensity Distribution

distribution (NLID) is shown for individual fields in Figure 1b and for fields taken together in Figure 1c.* The accumulative NLID gives a close-to-uniform "painted" scene on the vidicon with additive overlap between consecutive LED centers in adjacent fields. The resulting region between LED centers is an interpolation between adjacent LEDs.

A line of video read from the vidicon should correspond to the output of a single LED. This assumption is extended to include the weighted sum of consecutive LEDs as is the case with 2:1 interlace scanning in the FLIR. The only constraint is that the NLID of the LED array be approximately uniform from top to bottom. For example, consider the thermal scene in Figure 3a. Assuming cold is 0 and hot is 1, Figure 2b shows the video generated by the LED for the even and odd fields. With about 400 active video lines sampled from the vidicon every field, there are five equally spaced sampled lines between each of the 80 LEDs. The approximate video from the vidicon scanning in Figure 3 is shown in Figure 4. Figure 4a shows the detector and line scanning format; Figure 4b shows the video waveforms. It can be seen that synthetic DC restoration is still usable on this video since Assumptions 1 to 3 are still satisfied.

In conclusion, for scenes satisfying Assumptions 1 to 3 and with proper vidicon alignment, the synthetic DC restoration method should work well with the common module FLIR.

---

*To give an exact representation requires a detailed study of the FLIR optics. It is thought that this is a fair representation of the NLID.

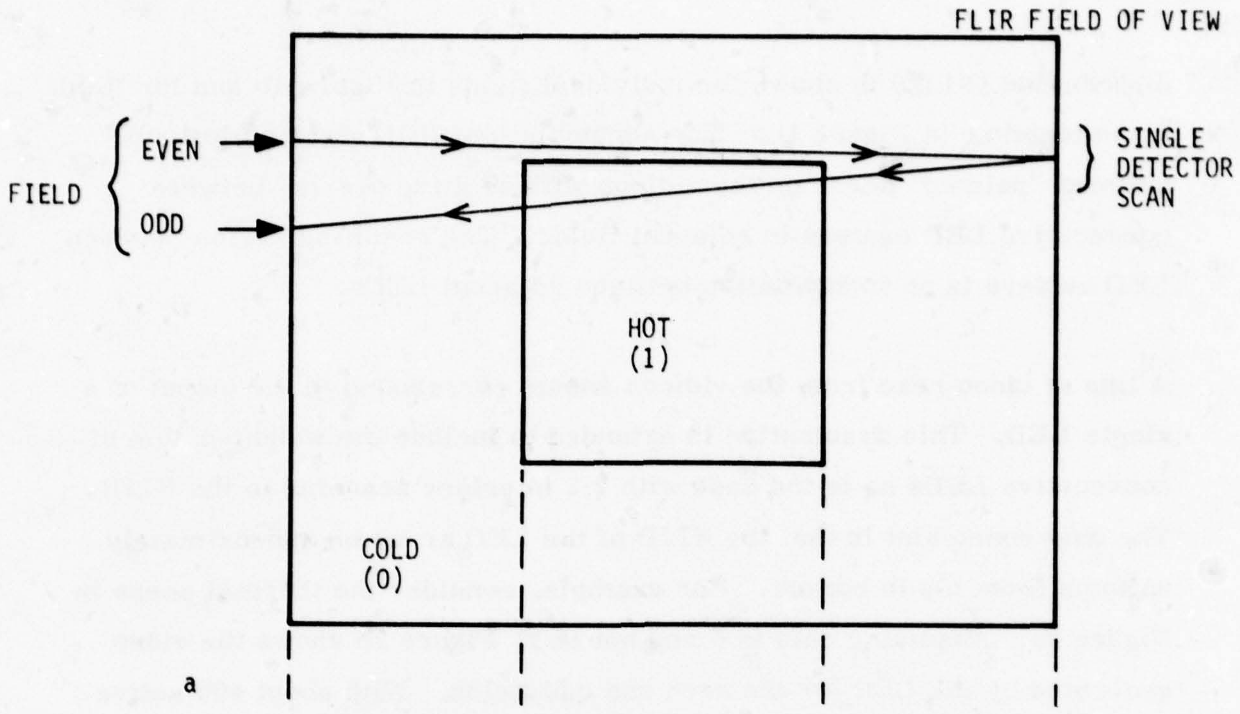


Figure 3a. Sample Thermal Scene and Detector Scan

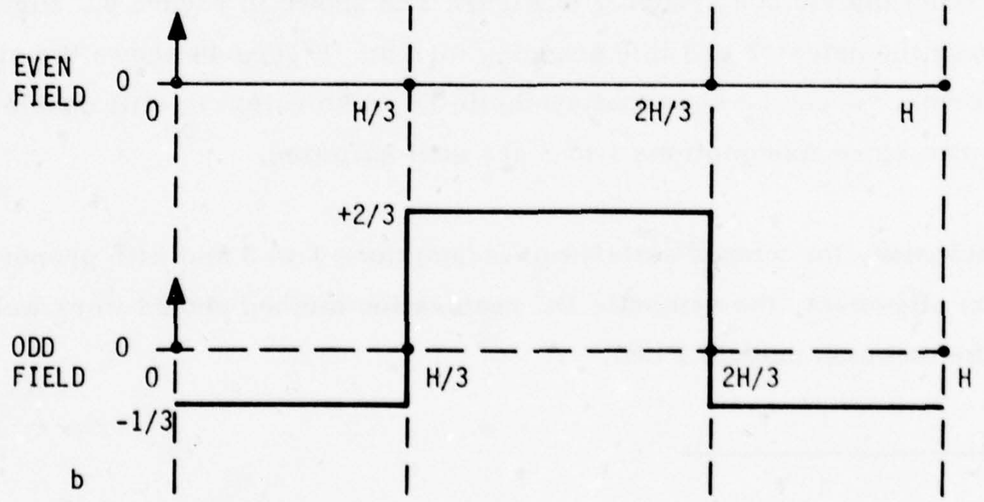


Figure 3b. Single Detector Output to LED Driver for Consecutive Fields

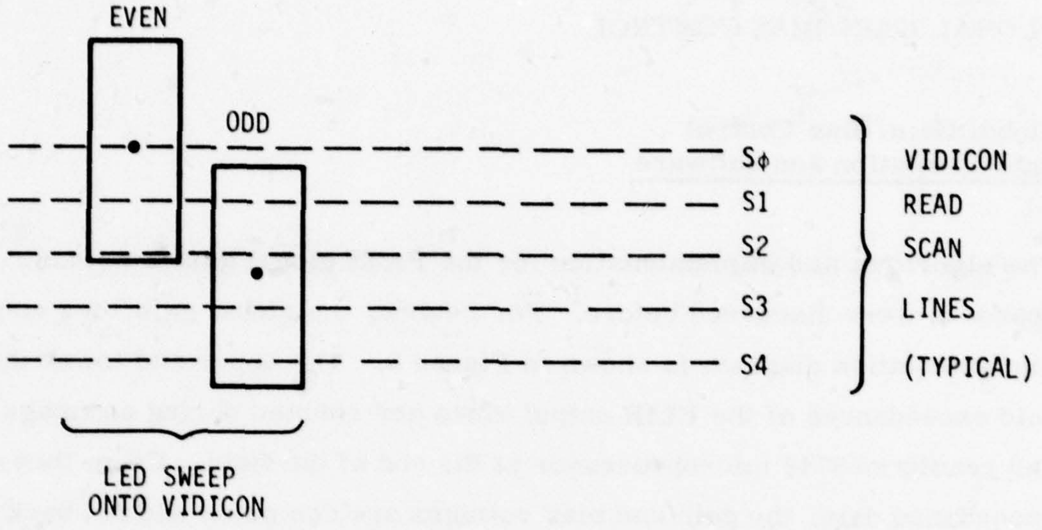


Figure 4a. Vidicon Scanning of LED Sweep

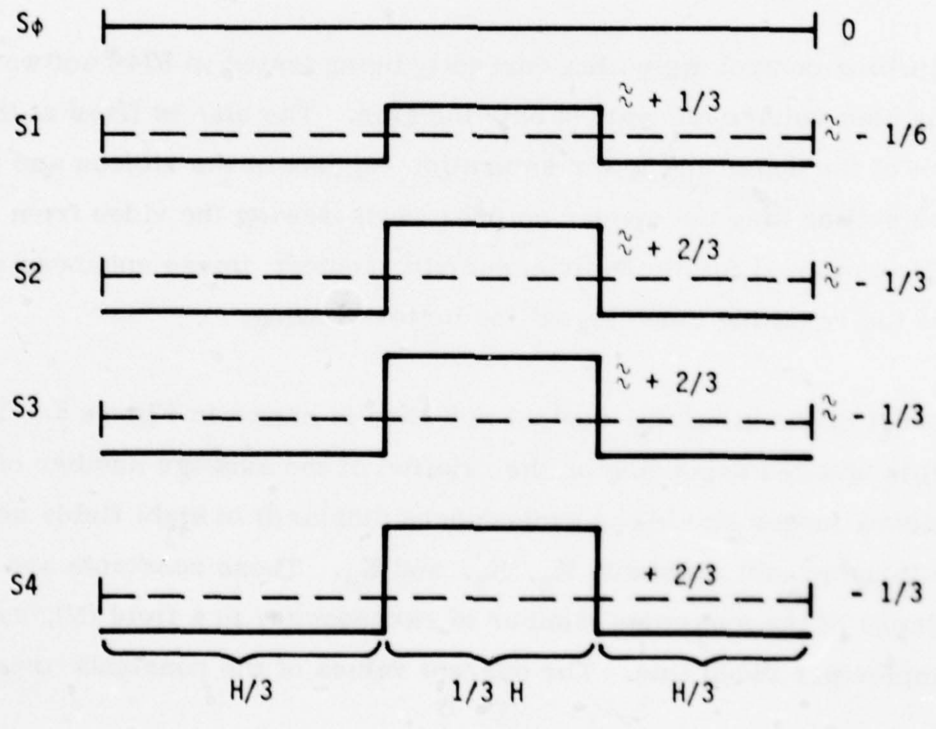


Figure 4b. Vidicon Video Format

GLOBAL GAIN/BIAS CONTROL

Global Gain/Bias Control Implementation and Software

The algorithm and implementation for the PATS global gain/bias control approach were discussed before.[1] For review, the global gain/bias control implementation diagram is shown in Figure 5. The upper and lower thresh-hold exceedances of the FLIR output video are counted during an image field and readby an 8748 microprocessor at the end of the field. From these exceedance data, the gain and bias voltages are computed and fed back to the post-amplifier stages in the FLIR. The algorithm to do the control is in software in the microprocessor.

The gain/bias control algorithm currently being tested in 8748 software fixes the bias voltage and varies only the gain. The bias is fixed at the midpoint of the upper and lower saturation regions of the vidicon and though this bias setting may not appear optimal when viewing the video from the FLIR, it is optimal for the vidicon and electronics; image enhancement will improve the resulting video signal for better viewing.

A flowchart of the gain/bias control software is shown in Figure 6. The gain (G) is updated depending on the relation of the average number of exceedances (upper and lower exceedances summed) in eight fields of data (N_T) to three preset constants K_0, K_1, and K_2. These constants are percentages of the maximum number of exceedances in a field (N), assuming 512 samples per video line. The current values of the constants are:

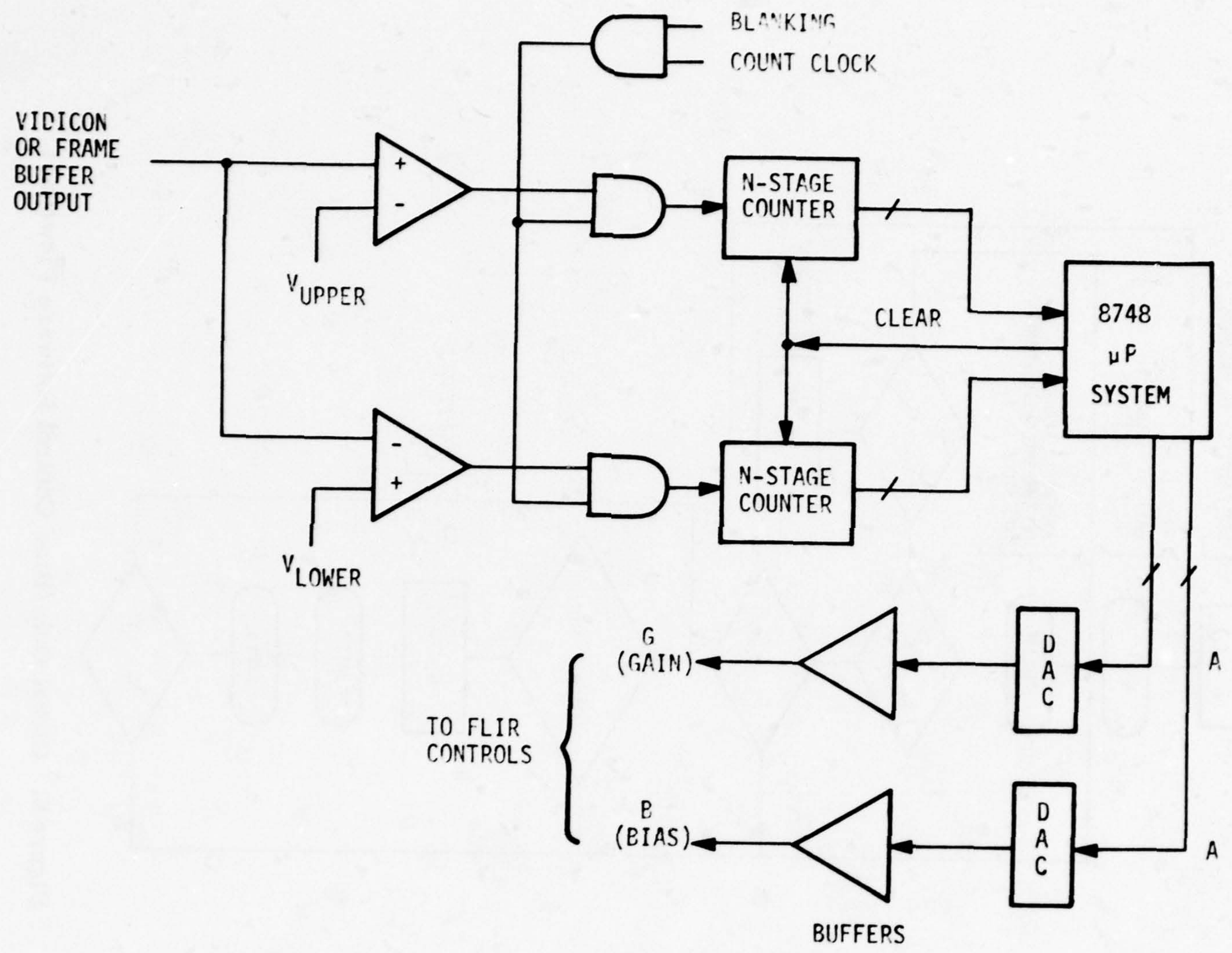


Figure 5. Global Gain/Bias Control Unit

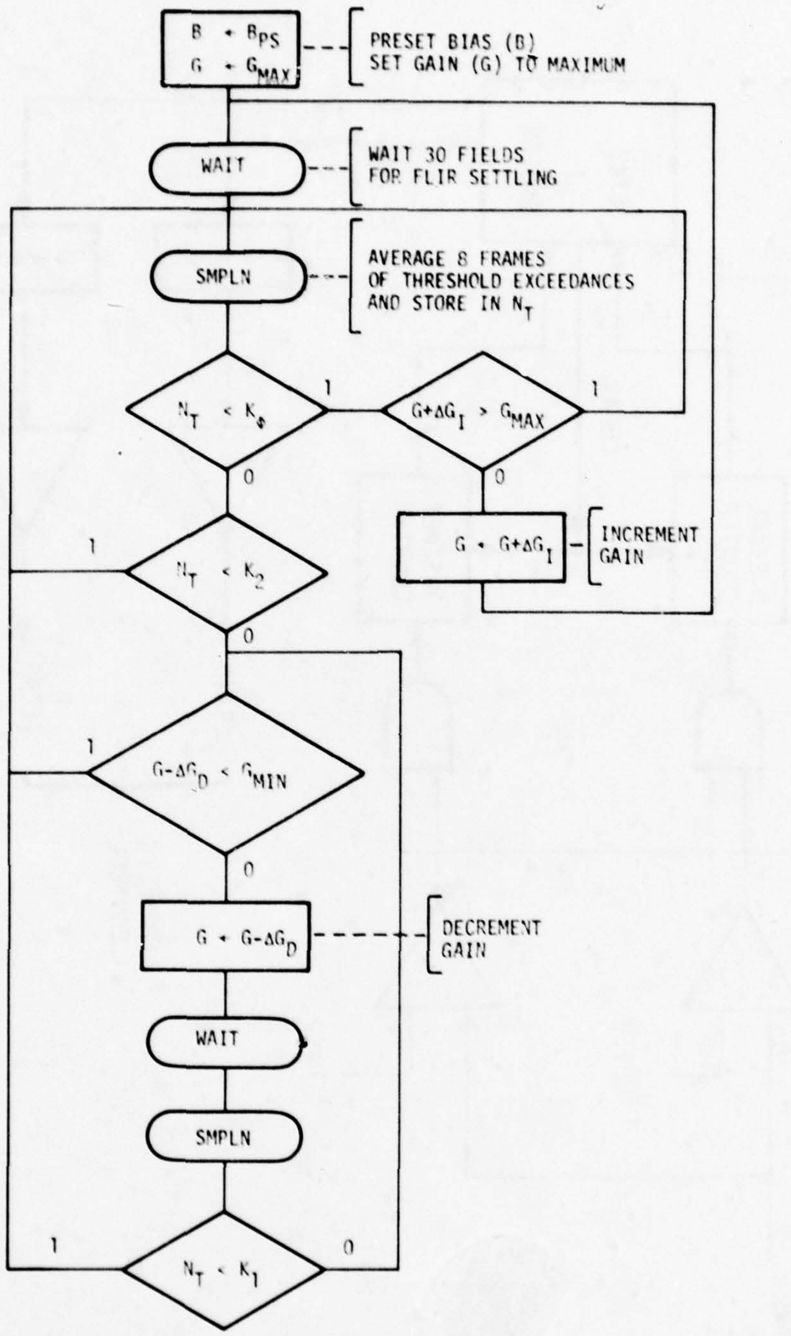


Figure 6. Global Gain/Bias Control Software Flowchart

$$K_0 = 0.002N$$

$$K_1 = 0.01N \qquad N \cong 20,000$$

$$K_2 = 0.03N$$

The gain is decreased gradually by an amount ΔG_D if N_T exceeds K_2. Decrement occurs until N_T falls below K_1 or reaches the minimum gain (G_{min}). If N_T falls below K_0, the gain is increased gradually by an amount ΔG_I until N_T goes above K_0 or the maximum gain (G_{max}) is reached. The quantities ΔG_D and ΔG_I are fixed with ΔG_D greater than ΔG_I. The increment values are expressed as a percent of the total gain adjustment range and have the following values:

$$\Delta G_D = 4\%$$

$$\Delta G_I = 2\%$$

These small values of ΔG_D and ΔG_I prevent oscillation but result in slow adjustment of the gain. The adjustment is slow because the settling time of the FLIR output to a step change in gain control voltage is about 0.5 seconds, or 30 fields. As a result of this and the sampling of eight frames of threshhold data, nearly 0.7 seconds of time is lost per gain update.

As an example of this, if the gain had to be decreased by 24 percent, the time required to do so would be about 4 seconds. To skirt this problem, the value of ΔG_D can be varied by increasing its value when N_T is larger. Presently, the value of ΔG_I must remain fixed because no information is available if the scene falls below the thresholds (N_T zero). The addition of scene standard deviation as an adjustment parameter for low contrast scenes is being considered to allow for a variable ΔG_I.

Global Gain/Bias Testing

The gain/bias control software is currently being checked out and has been integrated with MODFLIR. It is expected that the algorithm will work well on slowly varying scenes.

Modifications may be made to include scene standard deviation as a contrast adjustment parameter.

LOCAL AREA GAIN/BRIGHTNESS CONTROL

The image enhancement local area gain/brightness control (LAGBC) algorithm, implementation, and results on sample imagery were given previously.[1]

SECTION 3

TARGET SCREENING ALGORITHM SIMULATION

This section covers the progress in this reporting period on the target screener computer simulation, to include:

- Data base preparation
- Object extraction (segmentation) examples
- Feature analysis and selection
- Detection and recognition classifier methodology and prototype design
- Performance of the simulation on the test data base
- Interframe analysis results

SUMMARY

The Night Vision Laboratory (NVL) data base was expanded to 385 digitized frames. Modifications were made to the threshold computations and the interval generation scheme. A selected set of 110 frames was processed through the entire simulation. The measured features were evaluated for clutter discrimination and target classification. A preliminary set of features was selected and used to design a prototype clutter rejection classifier and a classifier to discriminate between tanks, APCs, and other targets. This classifier was tested on the training data and it performed satisfactorily. A set of eight independent FLIR frames from a sequence (one second apart) with moving targets was processed through this classifier and the results are used to illustrate the powerful interframe analysis approach.

DATA BASE PREPARATION

As we reported in the First Quarterly Progress Report,[1] our primary PATS FLIR data base is derived from the Fort Polk FLIR videotape supplied by NVL. The "NVL imagery" was used because it contains tanks and APCs at most aspect angles and reasonable ranges. This day and night imagery was acquired from a platform-mounted FLIR and is, therefore, appropriate for the low elevation angles encountered in tank battle and helicopter pop-up mission scenarios. In this reporting period we digitized 285 more subframes of imagery from this videotape to complement the NVL frames (100) digitized in the last period in an effort to complete representation of all aspect angles of tanks and APCs. Table 1 summarizes the membership of the NVL data base that now contains 385 digitized and annotated frames. As seen from Table 1, we still do not have a complete representation of all aspects of APCs. This is a reflection of the available aspect angles in the videotape. Figure 7 is a collection of sample frames from this data base. It illustrates the sensor and scene characteristics and shows how diverse target signatures can be under different imaging conditions.

IMAGE SEGMENTATION

A few changes were made to the segmentation scheme reported in the PATS January 1978 Quarterly Progress Report and are described below.

TABLE 1. SUMMARY OF THE DIGITIZED FLIR DATA BASE (NVL Data)

Target Type	Aspect Angle (Average elevation angle $\simeq 0°$)								
	0	45	90	135	180	215	270	315	Total
Tank	134	140	62	0	208	97	37	95	773
APC	65	23	0	0	58	63	50	0	259

Total number of frames = 385

Thresholds

Instead of the two thresholded binary images--one thresholded by a positive threshold and the other by an "absolute value" threshold--we now have two binary images, one obtained by a positive threshold and the other by a negative threshold (about the background estimate). This scheme extracts separately components which are hotter and colder (than the background). It is preferable to merge these components later, following syntactic rules, than have an absolute value threshold, which does not have the global information while at the low levels of the segmentation scheme and therefore tends to merge, for example, cold clutter with a hot target.

Threshold Selection

The two intensity thresholds (hot T_H, and cold T_c) are now computed using a recursive line average (i.e., the standard deviation). For line n,

$$T_H(n) = K_H \sigma(n) \text{ and } T_c(n) = K_c \sigma(n)$$

Figure 7. Sample Digitized FLIR Imagery from the NVL Videotape

$\sigma(n)$ is given by a first order, recursive smoothed estimate of the standard deviation of previous lines:

$$\sigma(n) = (1 - \gamma)\frac{1}{N} \sum_{M} \left| I(n - 1, M) - b(n - 1, M) \right| + \gamma\sigma(n - 1)$$

$I(n-1,M)$ is the image intensity at the point $(n-1,M)$, and $b(n-1,M)$ is the background estimate at that point. γ determines the number of lines over which this averaging is done. $\gamma = 0.9$ was chosen after experimentation and gives an effective exponential weighting that decays to 0.1 in 22 lines. Besides being easy to implement, this way of computing the threshold retains the advantages of a "local" standard deviation estimate based only on the previous line and is insensitive to anomalous changes in the standard deviation from line to line. Thus, this variance estimate replaces the global estimate reported in the January 1978 PATS report.[1] An identical scheme was implemented for smoothing the edge standard deviation estimate for the edge threshold computation.

Interval Generation

The interval generation criterion reported in the January 1978 PATS report[1] is asymmetrical; that is, it depends on the order of the data presentation to the algorithm (right to left vs. left to right). This is because an interval is started when an edge and a bright sequence are encountered; the interval is terminated when the bright sequence is interrupted (according to a smoothing criterion). Therefore, if an edge is present at the leading end of a bright streak, an interval is generated, but if it is at the trailing end, no interval is generated. Figure 8a, b, c, and d illustrates this phenomenon where we see that the tank has pronounced trailing edges, but the leading edges are not consistent. This causes the missing intervals.

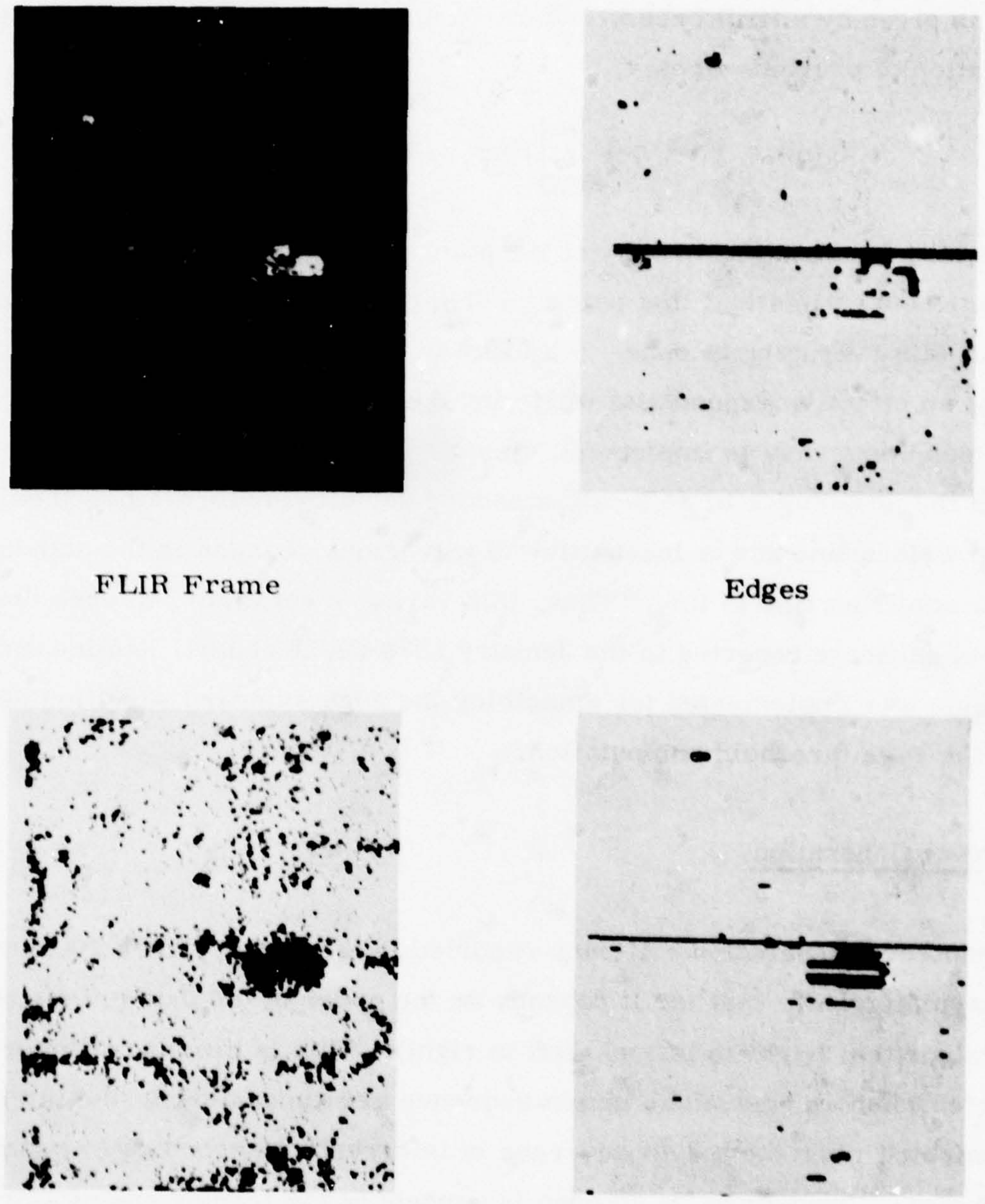


Figure 8. Result of One-Directional Interval Generation Criterion

To alleviate this problem, a bidirectional, interval generation criterion was conceived, simulated, and has proven robust. With this concept, an interval is started if there is a streak of brights (or separately, colds) and terminated when the bright ceases. But the interval is accepted if and only if there are either leading or trailing edges at the ends (see Figure 9). We call this validation of the interval and it is quite easy to implement. The result is identical to applying the previous interval generation criterion[1] twice to the scan line (forward and backward). This is shown in Figure 10, where the intervals are much more consistent than in Figure 8d.

Results of Segmentation of NVL Data Base

A subset of 110 images were processed through the segmentation simulation. Figure 11a, b, c, d, e, and f shows various stages of the segmentation process. Figure 11a is the original daytime FLIR frame containing an APC and three tanks. Figures 11b and c are the "hot" and "cold" thresholded binary images. Figure 11d is the thresholded binary edge image. The hot and cold binary images are separately associated with the edge image to yield intervals (on a scan line basis). Hot binarized points give rise to hot intervals and cold points give rise to cold intervals. Their identities are kept separate (Figure 11e). Both hot and cold intervals are then separately associated into hot and cold bins (object components). This is shown in Figure 11f. Here, the APC is entering the field of view and the middle tank is extracted as a hot component (the engine) and a cold component. Only the engines of the other two tanks are segmented as hot bins.

More examples of the object extraction process selected from the 110 frames processed appear in Figures 12 through 15.

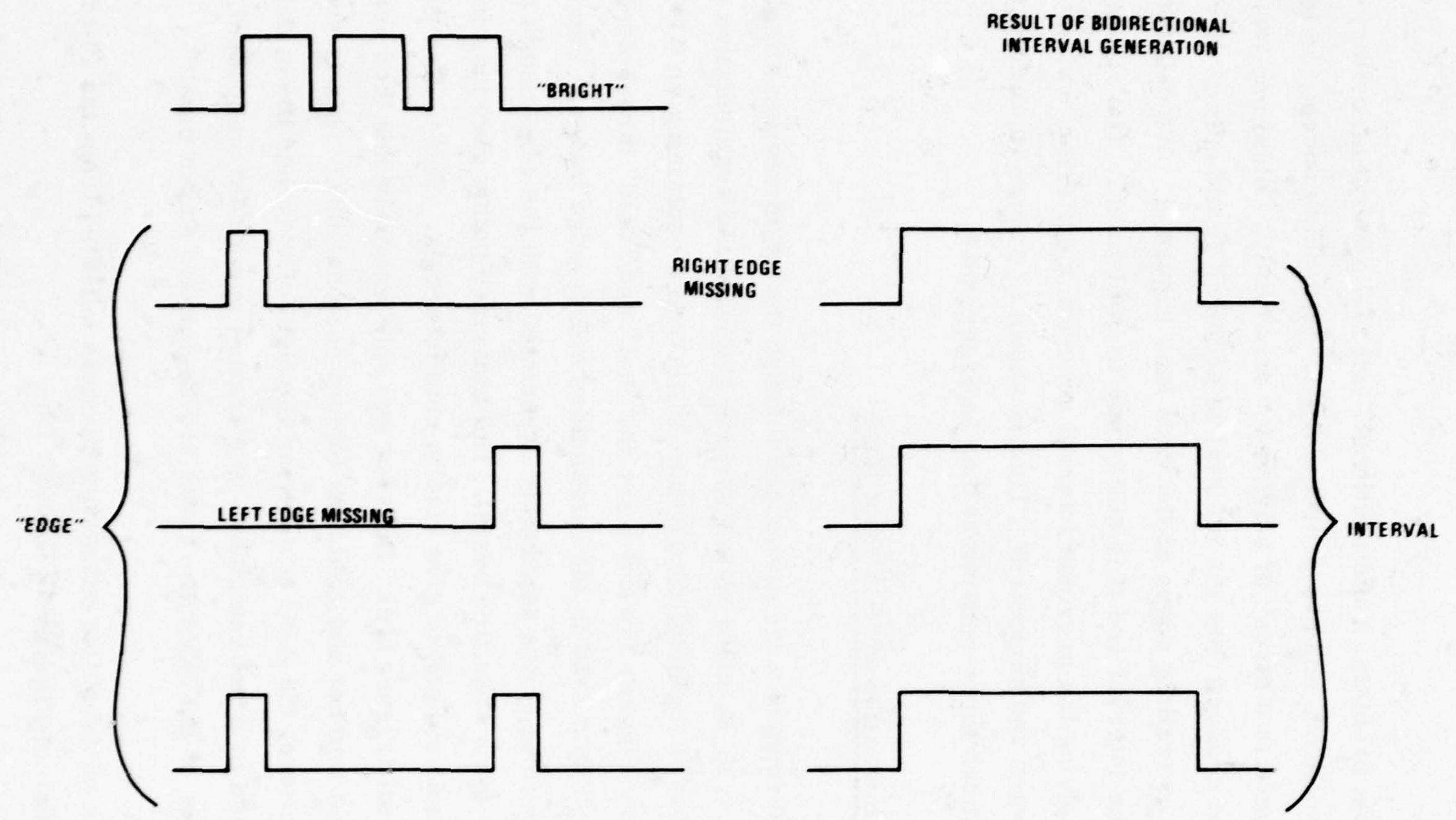


Figure 9. Symmetric Interval Generation Criterion

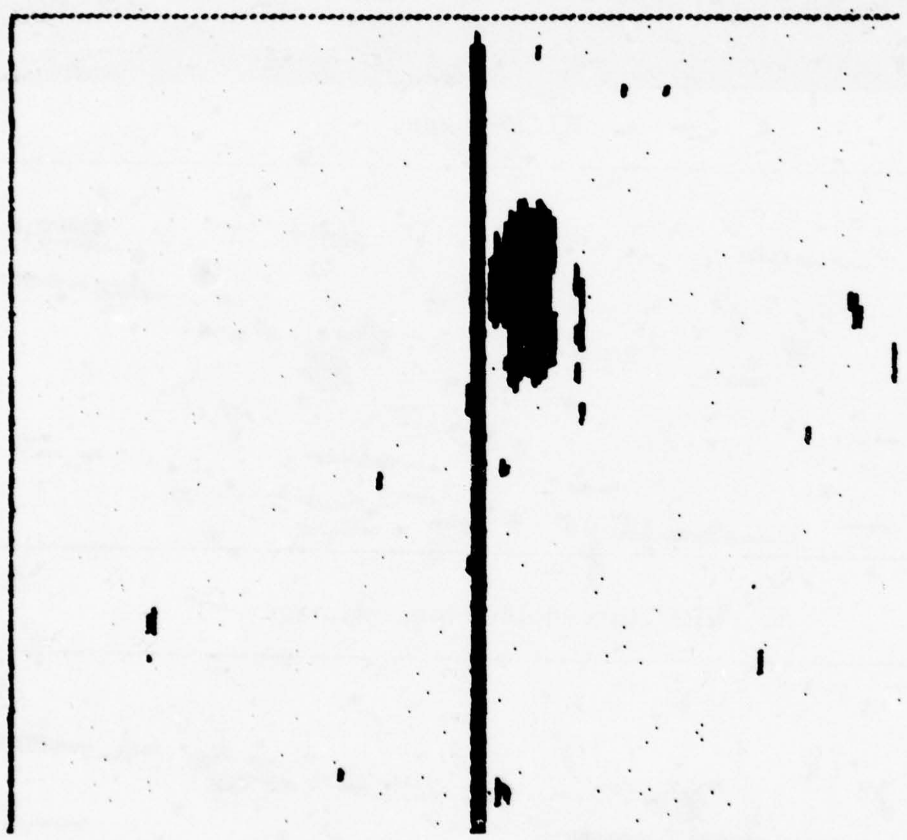
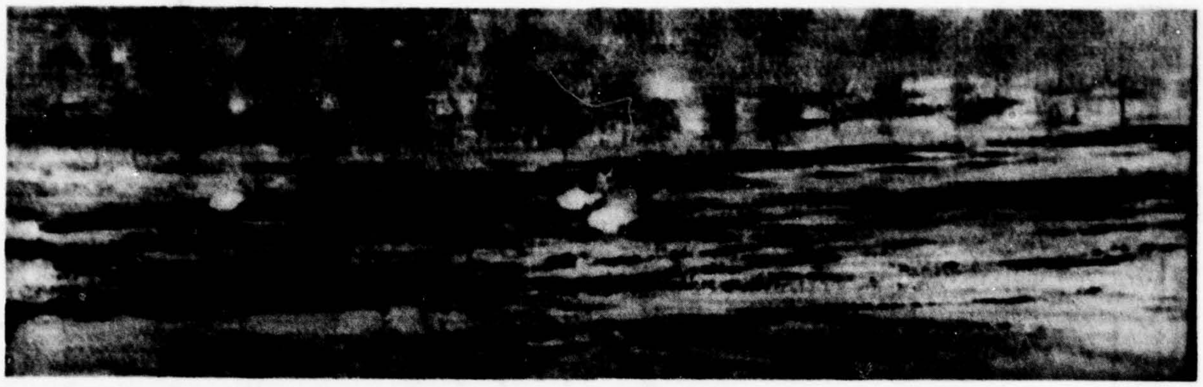
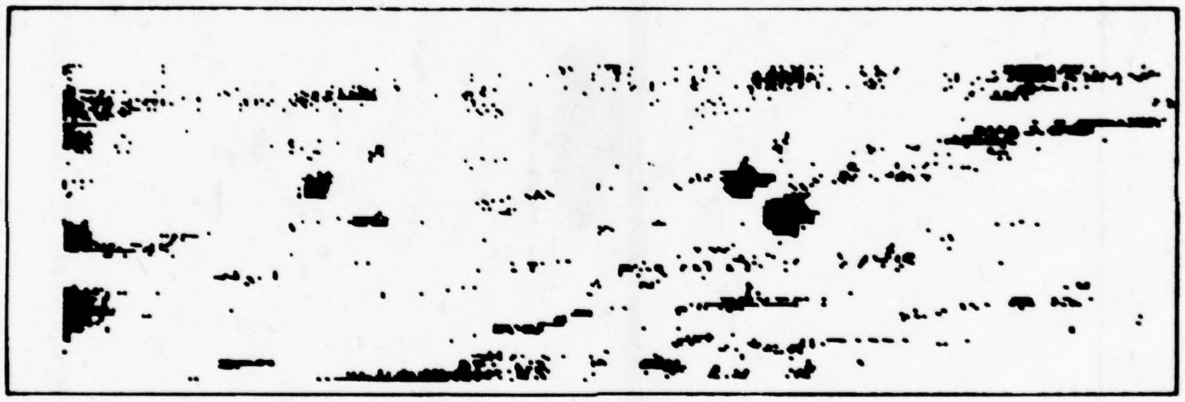


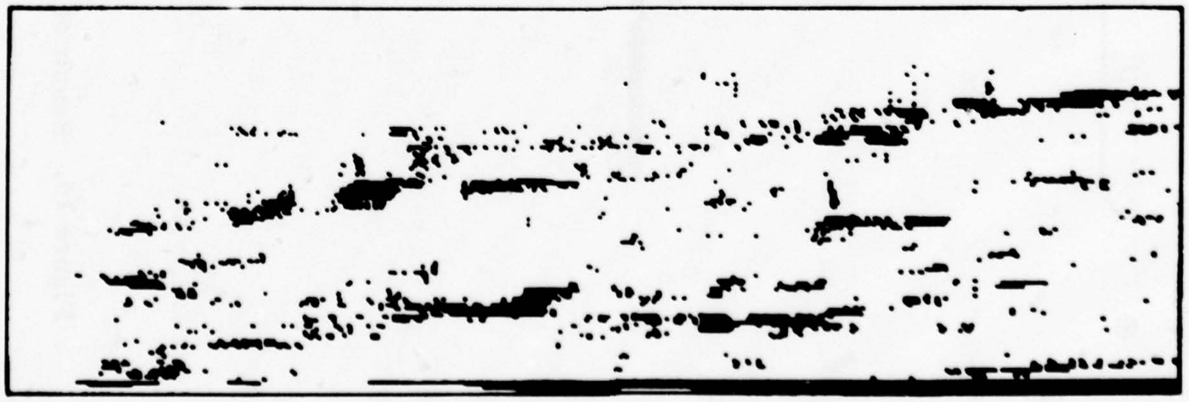
Figure 10. Result of Using Symmetric Criterion for Interval Generation



a. FLIR image

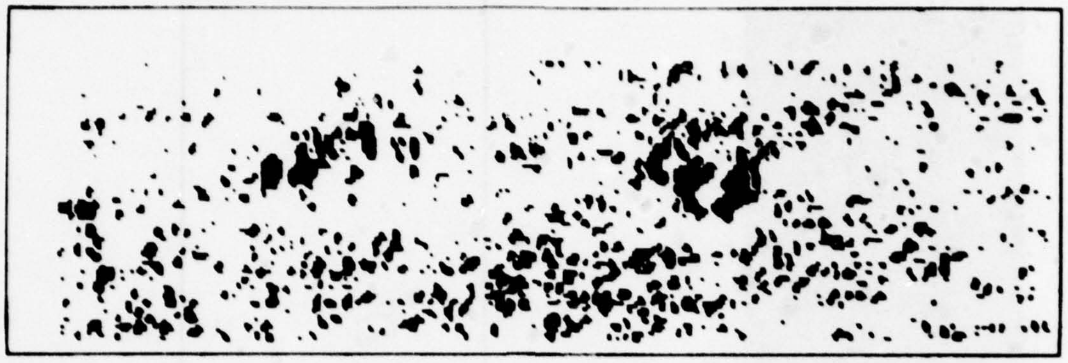


b. "Hot" thresholded binary image

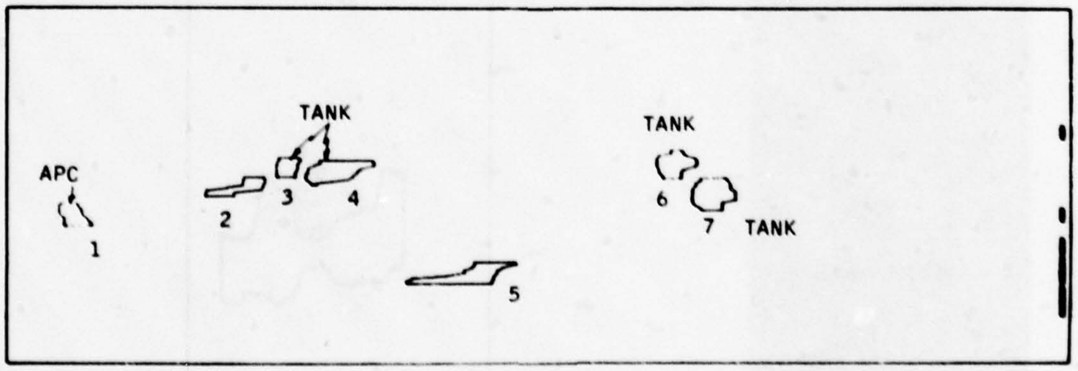


c. "Cold" thresholded binary image

Figure 11. Results of a Scene Adaptive Segmentation Scheme


d. Binary edge image

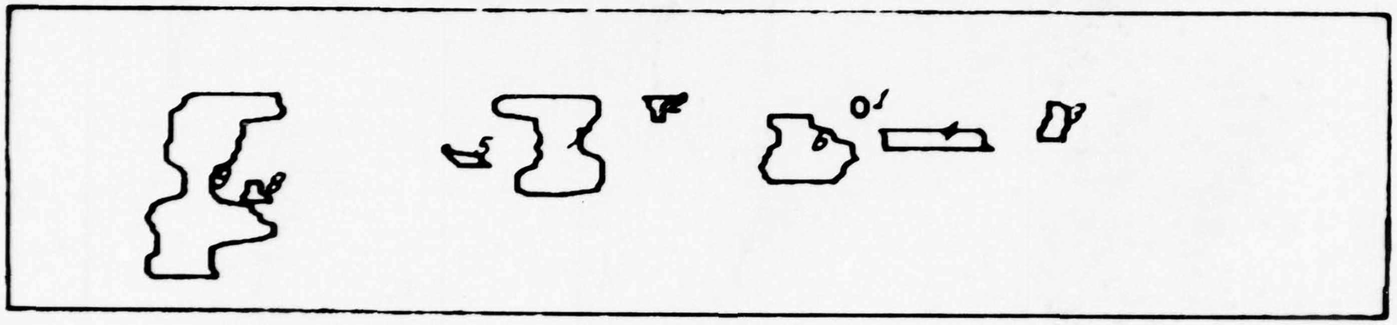
e. Object intervals



f. Extracted object components

Figure 11. Results of a Scene Adaptive Segmentation Scheme (concluded)

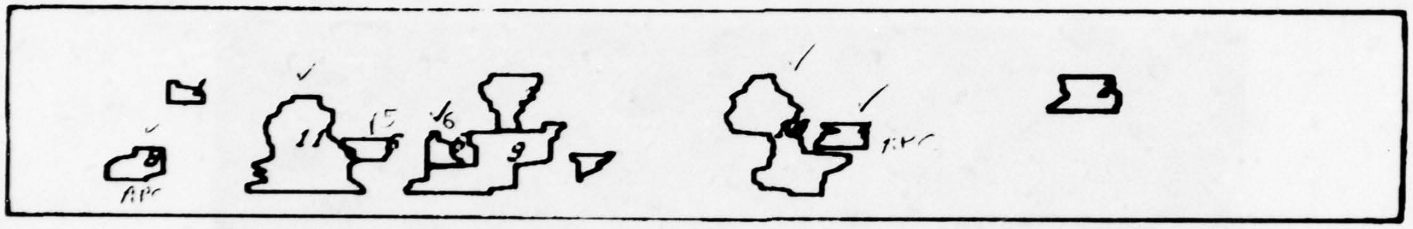
a. FLIR scene



b. Objects extracted

Figure 12. Examples of Object Extraction

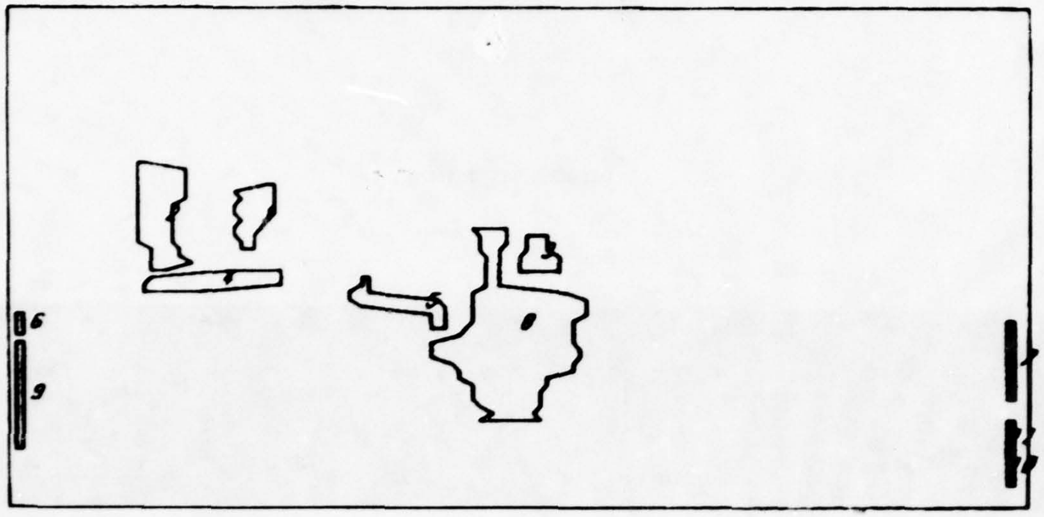
a. FLIR scene



b. Objects extracted

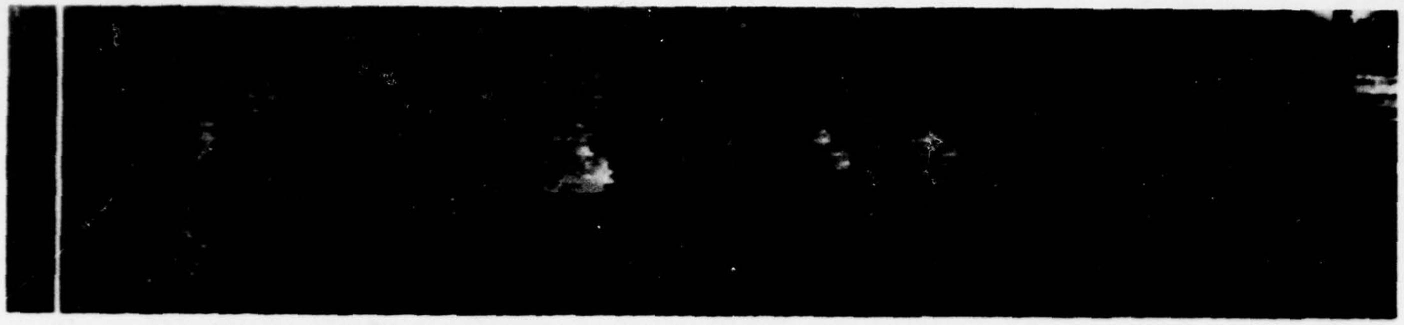
Figure 13. Examples of Object Extraction

a. FLIR scene

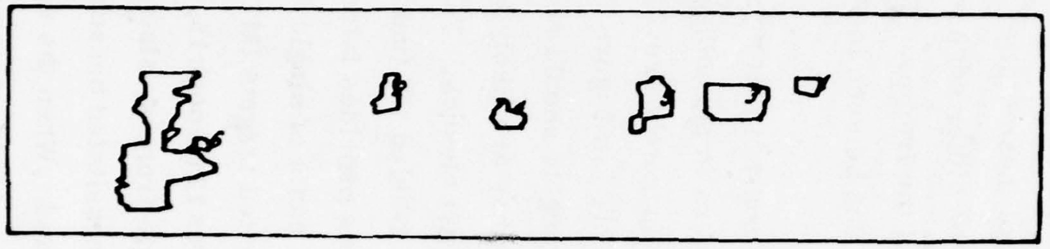


b. Objects extracted

Figure 14. Examples of Object Extraction



a. FLIR scene



b. Objects extracted

Figure 15. Examples of Object Extraction

These 110 images resulted in a total of 1,746 bins or extracted objects. Ground truth was subsequently provided manually for each object by painstaking examination of the segmentation output, the video disk image, and the videotape. This resulted in 393 targets and 1,353 clutter objects. A large number (810) of these clutter objects occurred because of a digitization characteristic on either edge of the image--the zero video portion was digitized on some of the frames. This gave rise to a number of small clutter bins. This can be seen in Figure 11f on the sides of the plot.

Figure 16 is a dendrogram that summarizes the ground truth on the extracted objects. The numbers in parentheses denote the membership of each class. Ground truth was separately provided to hot and cold (relative to the background) objects. "MT" in Figure 16 means "missegmented" targets; in our opinion these targets should not be used in training the recognition classifier because they are likely to be very different from the paradigms for the various target classes. Targets which were truly unidentifiable by the observer were labeled UT (unidentifiable). We must stress again that the ground truth was provided interactively with the help of the videotape and not just on the basis of single frames. Therefore, the number of improperly segmented targets (MTs) should be viewed in this context. The examples in Figures 12 through 15, where some of the targets are barely visible on the single frames, also support this point. The hot spot classes for tanks were necessitated because occasionally only the engines of the tanks were extracted. When the whole tank was extracted, it was placed in a different class. This was an attempt to recognize the tanks by the hotspots alone when only hotspots were visible.

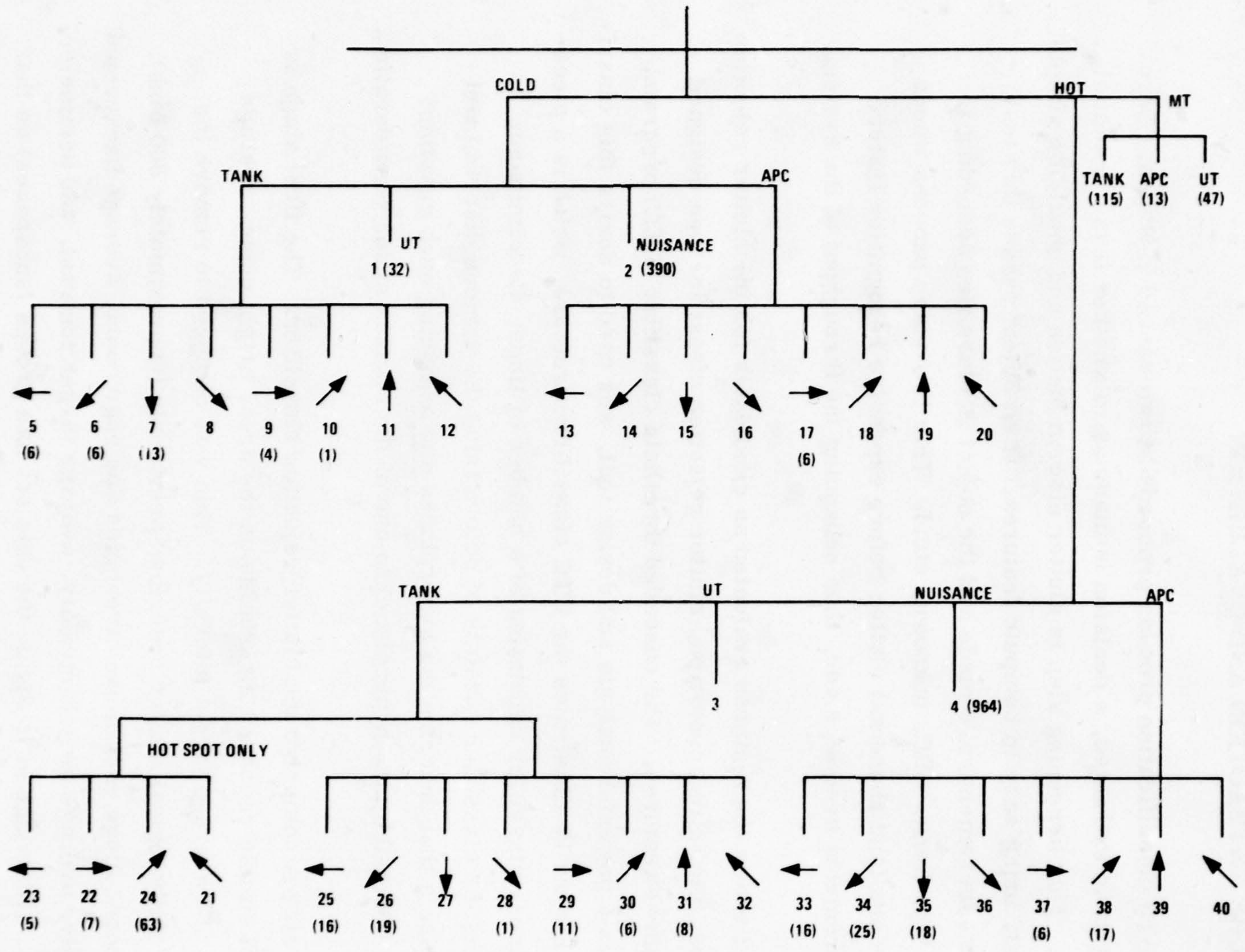


Figure 16. Ground Truth Distribution of Segmented Objects

FEATURE EXTRACTION AND EVALUATION

The PATS classification process proceeds in two cascaded stages. Once the object is extracted, a decision is made as to whether it is a potential target. This screening step is clutter discrimination and should be simple and fast, using easy to compute features. If an object passes this test, more measurements are made and the object is classified according to target type (tank, APC, unknown, etc.). This two-stage process weeds out obvious (and abundant) clutter before expensive recognition feature measurements need be made, thus enhancing the throughput of the system.

Table 2 shows the features evaluated as candidates for the clutter rejection process. An initial prototype, clutter rejection classifier was designed using these features. The cascaded threshold classifier (CTC) program, which is a powerful analysis and design tool, was used to derive this classifier. Figure 17 illustrates the CTC classifier structure, which is a piecewise linear classifier consisting of a number of linear discriminants. Training the classifier consists of determining the various discriminant coefficients that describe the hyperplanes <u>and</u> assigning each resultant region (formed by the hyperplane) to one of the classes under consideration.

Three stages comprise the clutter rejection classifier. The first stage is a set of simple cascaded thresholds on the area, bright count, average width, and average target intensity. This was designed to remove the "easy" pathological clutter described previously (approximately 800 bins). The second stage uses these thresholds and edge count, average background intensity, average target intensity, average target contrast, and perimeter/ $\sqrt{\text{area}}$. This stage is to divide the data set into groups (subspaces) so that

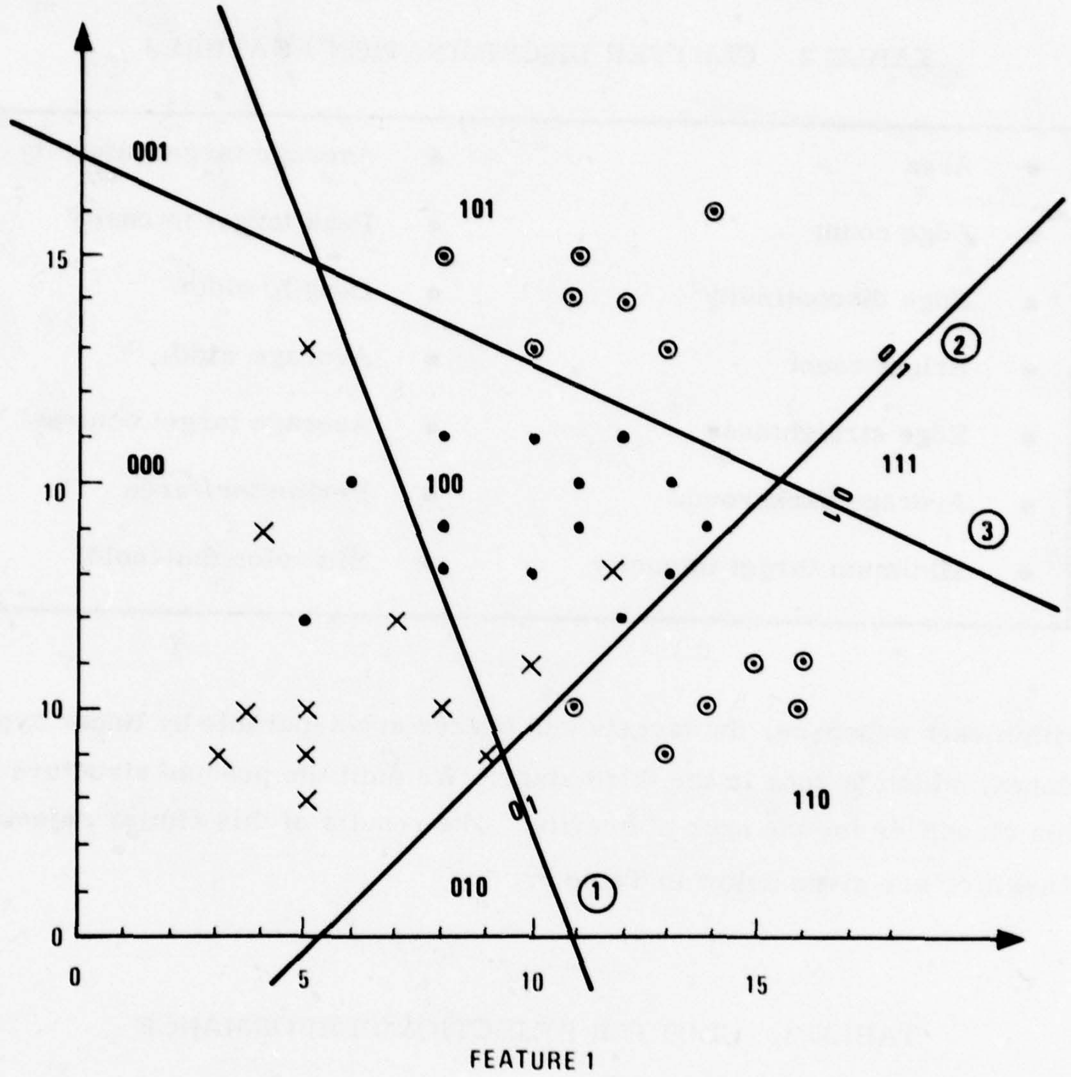
TABLE 2. CLUTTER DISCRIMINATION FEATURES

● Area	● Average target intensity
● Edge count	● Peak target intensity
● Edge discontinuity	● Length/width
● Bright count	● Average width
● Edge straightness	● Average target contrast
● Average background	● Perimeter/$\sqrt{\text{area}}$
● Minimum target intensity	● Bin color (hot/cold)

within each subspace, the targets and clutter are separable by linear hyper-planes, which is done in the third stage. We omit the precise structure of this classifier for the sake of brevity. The results of this clutter rejection classifier are given below in Table 3.

TABLE 3. CLUTTER REJECTION PERFORMANCE

	Reject	Accept
Clutter	1227	124
Target	70	315



	KEY	DECISION
CLASS 1	X	(000)
CLASS 2	●	(100)
CLASS 3	⊙	(101) U (110)

Figure 17. PATS Cascaded Thresholded Classifier Approach

The maximum number of vector products (discriminant computations) needed is a measure of the classifier computational complexity. The clutter rejection classifier, being a multistage classifier, poses different computational requirements depending on which branch of the classification tree one goes down. However, the maximum number of vector products needed for the deepest branch is 5, and the maximum number of features involved at any given discriminant computation is 10.

We have deliberately biased the clutter rejection classifier in favor of targets because we do not wish to miss targets at this early stage. Moreover, any accepted clutter may yet be removed in the subsequent (recognition and interframe analysis) stages.

This classifier can be further optimized to minimize the number of features used (by further feature evaluation) and the number of stages of hierarchical discrimination. This will be done in the next reporting period. Further simplification of the classifier has the added benefit of making it more robust with respect to the training data.

RECOGNITION CLASSIFIER

The candidate recognition features--three classes of moment features and the Fourier boundary descriptors (FBD)--were described in the first interim report.[1] Each class of these features was evaluated to determine its discriminatory power with the well-segmented target bins.

The Fourier boundary descriptors proved to possess very little separation among the various target classes. Two-dimensional scatter plots of pairs of these features taken at a time showed that all the classes appeared to come from the same distribution. Figures 18 and 19 show scatter plots of two pairs of the Fourier amplitudes to illustrate this point. As a consequence, the FBD features were eliminated from further analysis.

The moment features proved to be of greater value for class discrimination, but there was significant correlation among similar moments from each class (boundary, silhouette, and intensity). Figure 20 shows an example of this correlation between μ_{20} (intensity) and μ_{20} (silhouette). Therefore, for the prototype classifier we chose the six intensity moments, the peak and average contrasts, and the length/width $(p+\alpha/2)$ features (Table 4). Because all moments $\mu_{p\alpha}$ are normalized by $(\mu_{20} + \mu_{02})$ (see PATS first interim report[1]), the normalized μ_{02} and μ_{02} are perfectly negatively correlated. Therefore, μ_{20} was omitted.

A prototype classifier was designed with these nine features using the cascaded threshold classifier programs. Eight target classes were defined (Table 5). Linear discriminants were computed to separate all pairs of classes (28 pairs). All the target vectors were projected onto directions orthogonal to each of these hyperplanes, and these one-dimensional projections were analyzed for class separation. Those that yielded the maximum separation (or clustering) of the various classes were chosen and thresholds were determined for each projection from this analysis. A total of six discriminants were chosen.

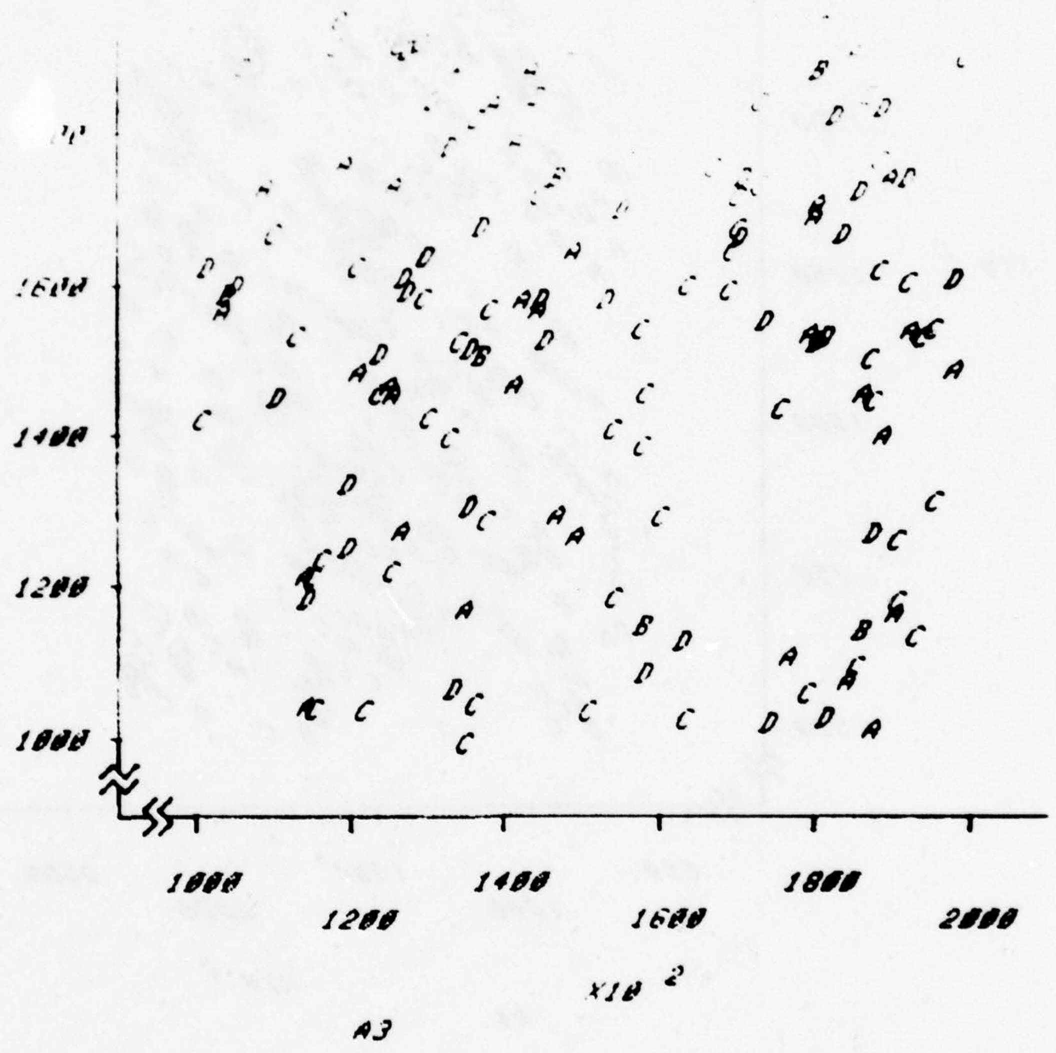


Figure 18. Fourier Amplitude Features (a_3, a_4) (Show no separation between target classes A, B, C, D)

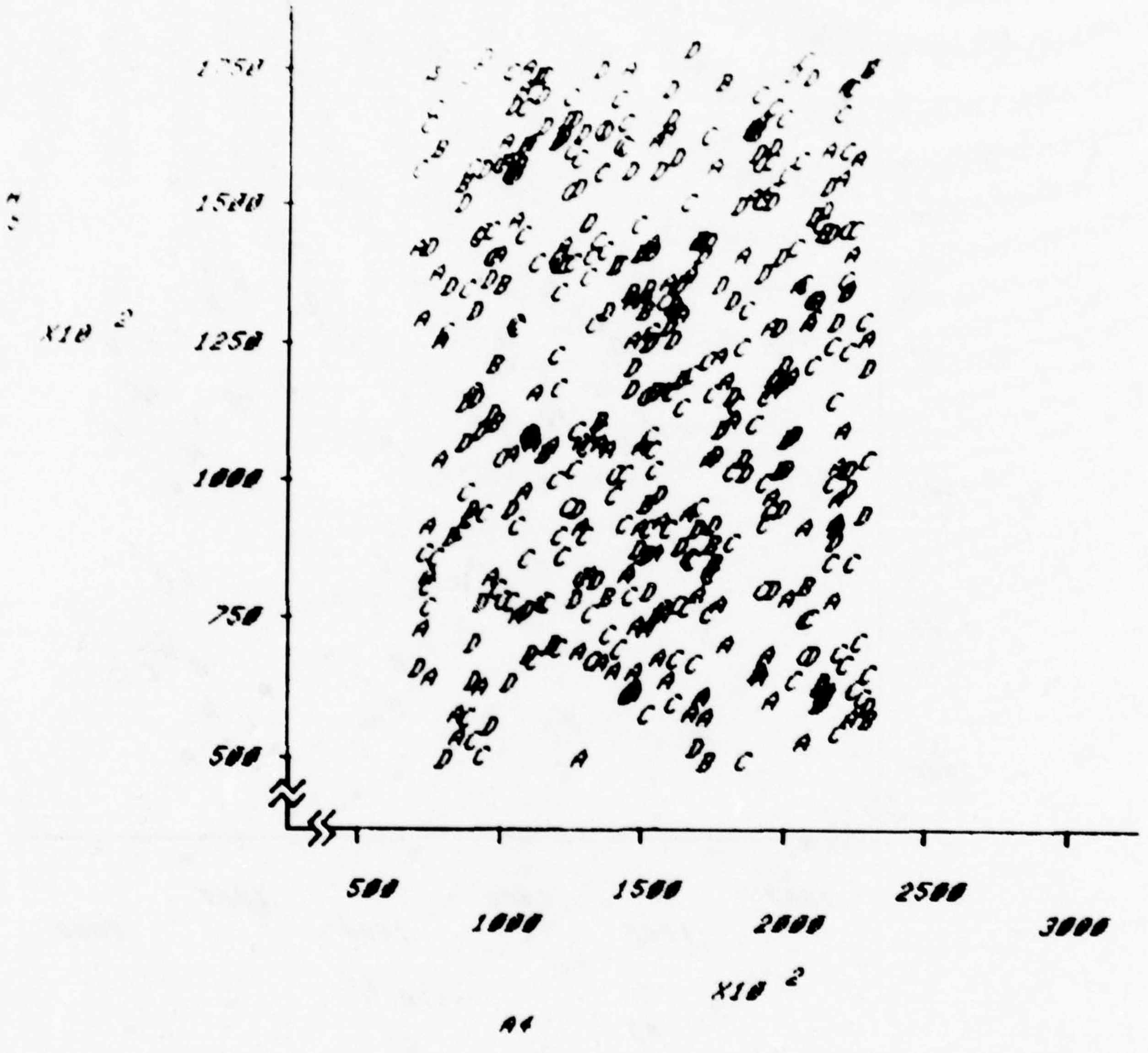


Figure 19. Fourier Amplitude Features (a_4, a_5) (Show no separation between target classes A, B, C, D)

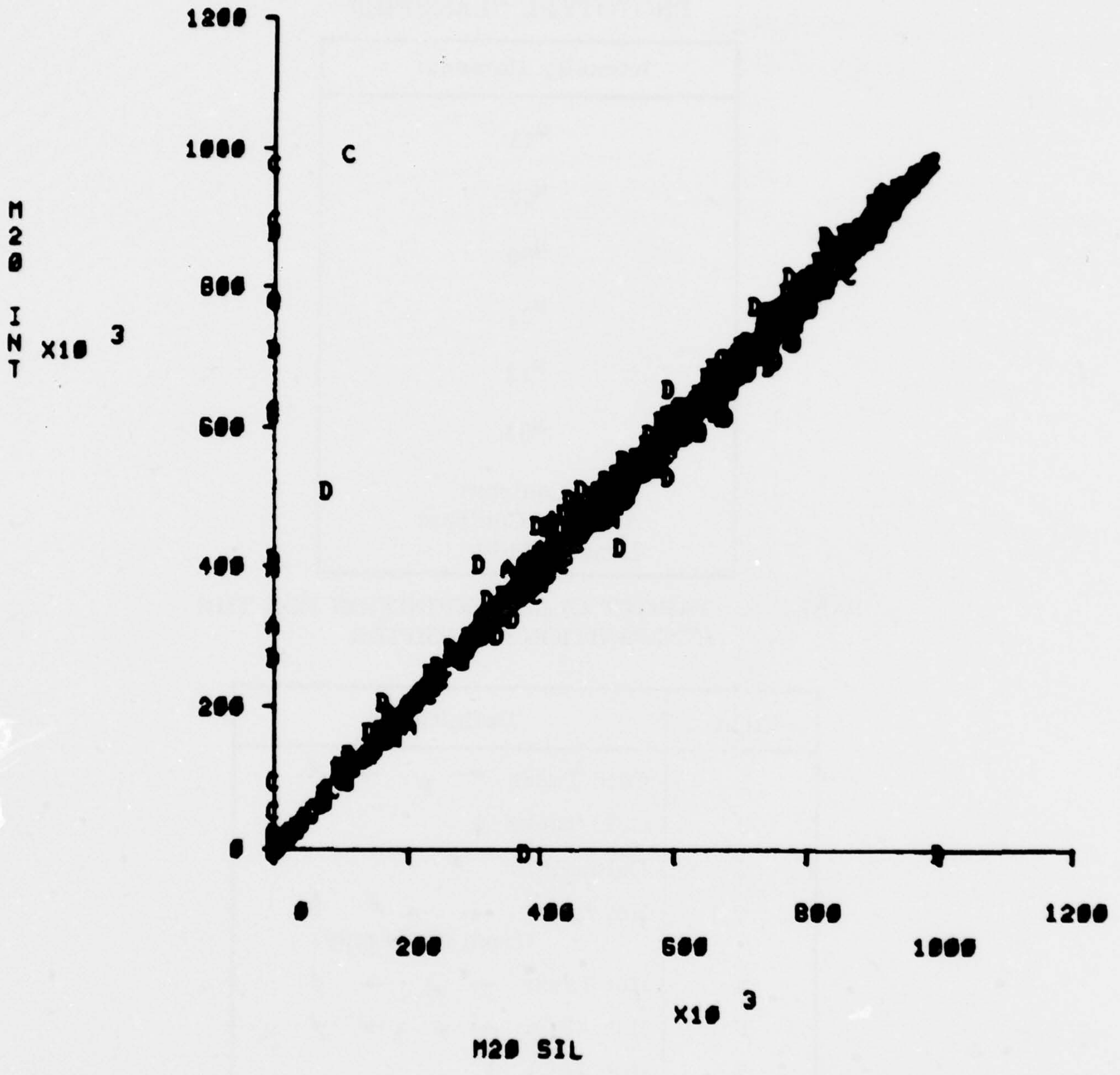


Figure 20. Scatter Plot of μ20 (intensity) vs. μ20 (silhouette) (Shows correlation of "like" moments from different moment classes)

TABLE 4. RECOGNITION FEATURES USED IN PROTOTYPE CLASSIFIER

Intensity Moments
μ_{11}
μ_{02}
μ_{30}
μ_{21}
μ_{12}
μ_{03}
Peak Contrast
Average Contrast
Length/Width

TABLE 5. TARGET CLASS DEFINITION FOR THE RECOGNITION CLASSIFIER

Class	Definition
1	Cold Tanks ← ↙ → ↗
2	Cold Tanks ↓
3	Cold APCs ↗
4	Hot Spots ← → ↗ ↑ (from tanks only)
5	Hot Tanks ← ↙ → ↗
6	Hot APCs ← ↙ ↘ → ↗
7	Hot APCs ↓
8	Cold UTs and Hot UTs

These six discriminants define, at most, 2^6 regions in the hyperspace (see Figure 17 for example). The next step defines the class membership of each region to determine how the training data were classified. This step simultaneously designs the classifier and yields an estimate of the classifier performance. This process was done by histogramming the training samples into the various regions and assigning each nonempty region to a class depending on the majority class of the target training samples that fell in the region.

Once the classifier was designed, all nonclutter samples (including the missegmented targets) were run through the classifier. Table 6 summarizes the classification results of the entire data set for both stages of classifications (clutter rejection and target classification). The number of false alarms is also shown (102 in 107 frames) and is within the design goal maximum of one per frame. As we will see later, interframe analysis will significantly reduce this rate even further. From this table, probability of detection equals 78 percent. Among the samples that were classified as tanks or APCs, tanks were correctly classified 69 percent of the time. The corresponding probability of correct recognition for APCs was 91 percent.

INTERFRAME ANALYSIS

To test the validity of interframe analysis for target classification, a sequence of FLIR frames one-third second apart were acquired on a video disk (Figure 21a). The sequence represents moving targets (APC and three tanks) with partial and complete occlusion (of the middle tank) and partial occlusion of the two tanks on the right of the field of view with each other. Eight frames (now one second apart) were selected from this sequence and independently processed through the segmentation and feature extraction simulation programs. Figure 22 shows a sequence of eight frames from Figures 21a and 21b segmented individually using the current PATS simulation. Note that the APC on the left is entering the field of view and is completely extracted in Frames 2 and 3. It is only partially segmented in Frames 4 and 5 and reappears as complete in Frames 6 and 8. In some frames, hot spots on the tanks are extracted because the colder areas of the tanks are lower contrast with respect to the background. The colder areas of the middle tank are also extracted in Frames 1, 2, 5, 6, and 8. This tank is partially occluded behind a tree in Frames 3 and 4 and completely occluded in Frame 5. This sequence also illustrates targets partially occluding one another (the two tanks being merged in Frames 6 and 8).

TABLE 6. PERFORMANCE ON TRAINING DATA

		Classified Objects				
		T	A	U	C	
Ground Truth	T	129	58	15	48	250
	A	5	50	1	31	87
	U	16	8	9	14	47
	C	59	43	11	1243	1356

102 False Alarms in 107 Frames

$P_D = 0.78$

T	=	Tank
A	=	APC
U	=	Unidentified targets
C	=	Clutter

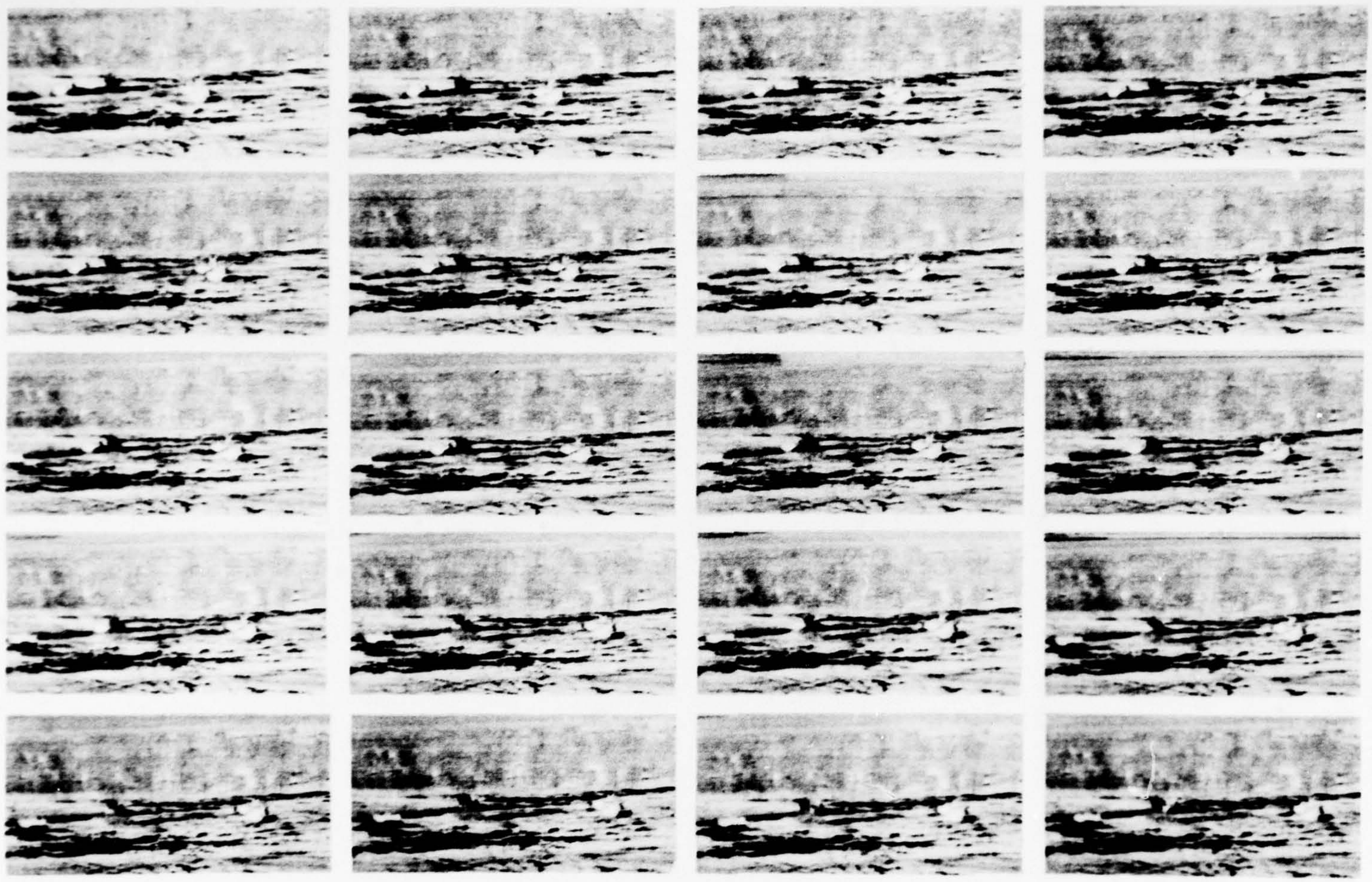


Figure 21a. An Example of a Sequence of Tactical Imagery

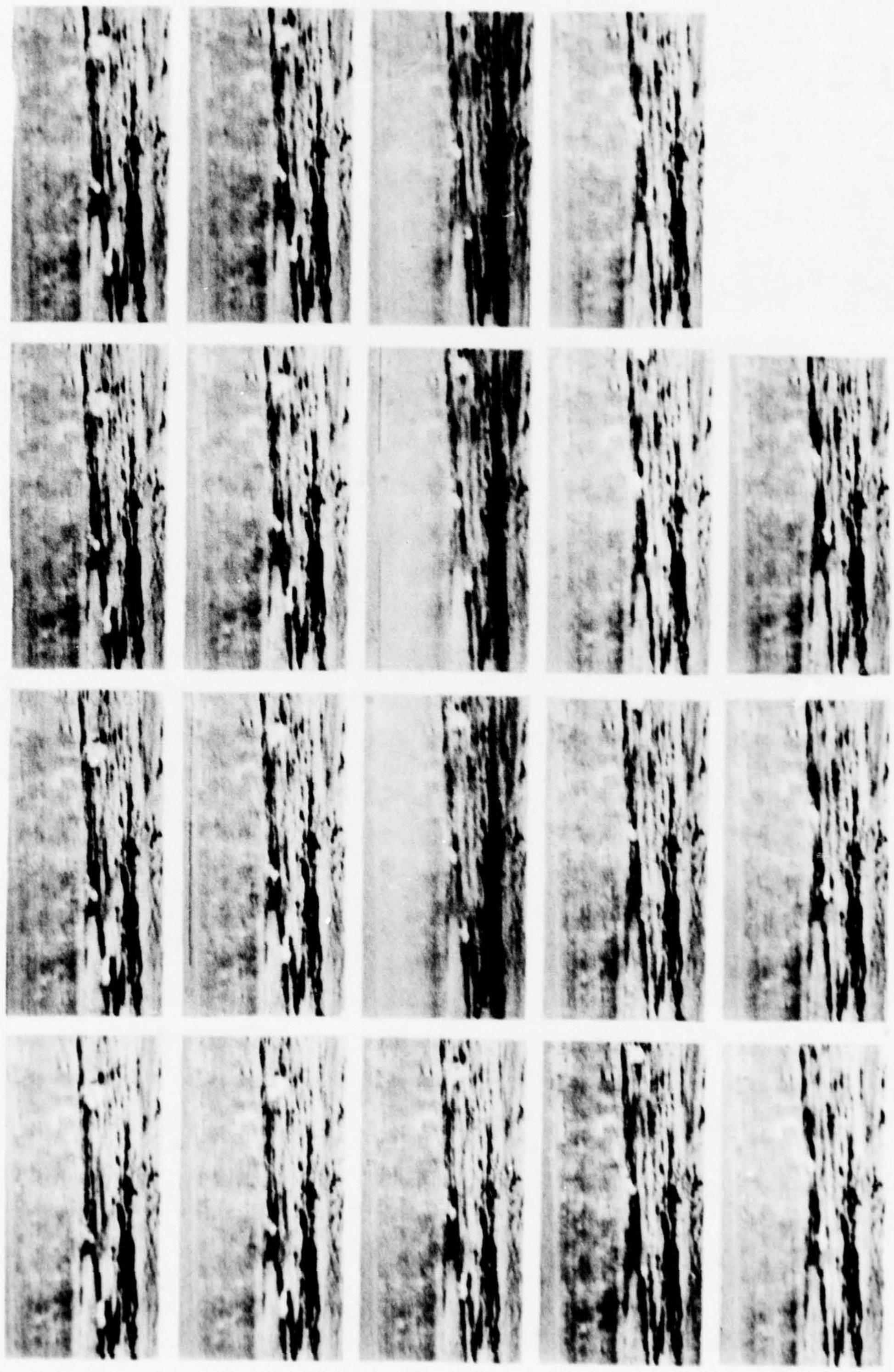


Figure 21b. An Example of a Sequence of Tactical Imagery (concluded)

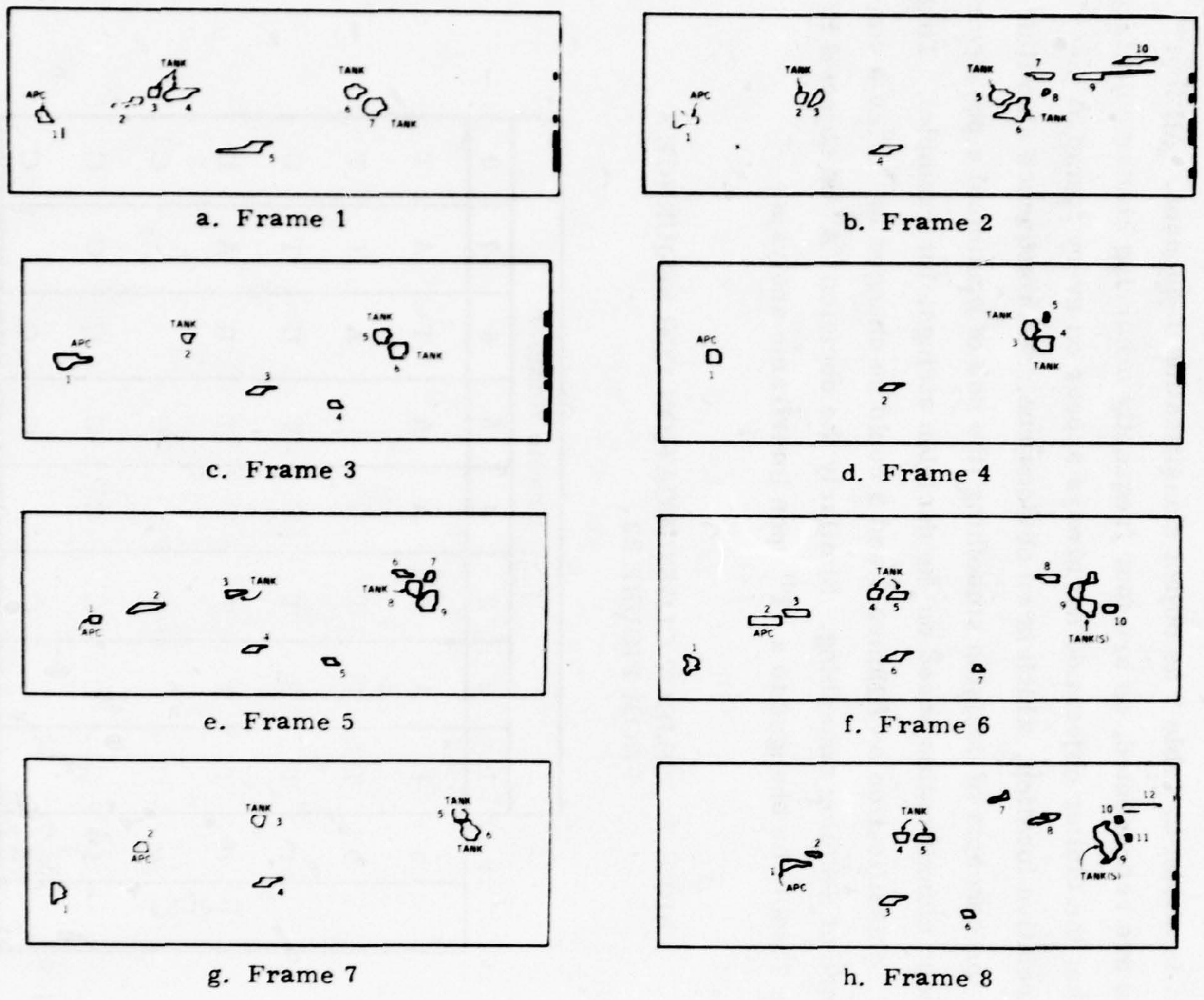


Figure 22. Result of Segmentation of FLIR Frame Sequence for Interframe Analysis

The feature vectors from these extracted objects were input to the clutter rejection and recognition classifiers. The result of this classifier for eight objects is shown in Table 7 as object classification sequences. All four targets are represented, as are four frequently occurring clutter objects. Note that the clutter objects do not always appear on every frame at the same relative location, which is an obvious cue. This sequence also illustrates the concepts of decision smoothing (the use of maximum a posteriori likelihood classification based on the decision strings, for example). Thus, the "U" classification on Frames 6 and 8 could be changed to "T" as a consequence of decision smoothing. Similarly the decision "A" on object d in Frame 7 would be changed to a "T" upon interframe analysis.

TABLE 7. OBJECT CLASSIFICATION FOR SEQUENCE FROM FIGURE 22

		Frame Number							
		1	2	3	4	5	6	7	8
Object	a	T	A	T	A	A	T	A	T
	b	T	T	T			A	T	T
	c	T	T	T	T	T	U	T	U
	d	T	T	T	T	T	U	A	U
	e	C				C			C
	f	C	C	C	C	C	C	C	C
	g			C		C	C		C

T = Target
C = Clutter

These rules for decision smoothing are based on 1) the statistics of the scene (the a priori probability of tanks, APCs, etc., in the scene); 2) the bias of the classifier (the probability that a tank is classified as an APC and vice versa); and 3) the assigned priority of the target class (how critical it is not to call an APC a tank). The first and third parameters depend on the operational scenario and can be specified by the tank commander, for example. The second parameter could be determined by processing sequences of imagery through the simulation.

However, we feel that a cost-effective approach to determining the classifer bias will be to use the PATS hardware (upon completion) to gather statistical data for the interframe decision smoothing algorithm. Processor 2 in PATS is eminently suitable for this task because it is easily programmable and can be interfaced to external peripherals for data transfer.

SECTION 4

SYSTEM DESIGN

HARDWARE DESIGN

The PATS hardware can be broken up into several functional units, as shown in Figure 23. These functional blocks, detailed later, are:

- Autothreshold--The automatic adaptation to varying contrast levels that produce three binary signals--hot, cold, and edge.
- Interval generation--The logical combining of the binary signals generated by autothreshold and a valid segmented object made up of several intervals.
- First level features--The real-time generation of specific values associated with each interval.
- Intensity data--The storage of digitized video data over the scene of interest.
- Computer system 1--Processes the first level features and intensity data for the detection and recognition of specified target types.
- Computer system 2--Provides for the diagnostic checkout of the system and determines the type of symbol to be generated.
- Symbol generation--The generation of a symbol specifying the target type recognized and the location in the scene. The symbol is added to the displayed video.

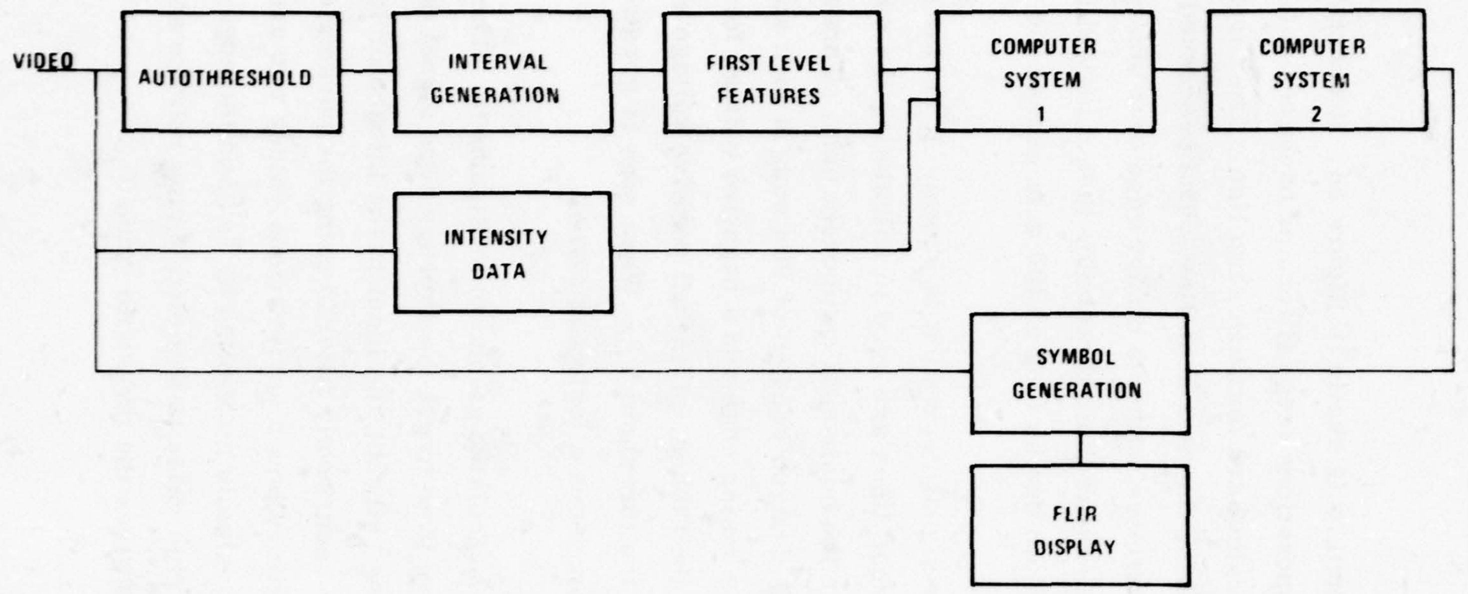


Figure 23. PATS Functional Units

Autothreshold

The autothreshold structure is shown in Figure 24. It consists of two different two-dimensional operations (edge filter and background filter). An adaptive threshold is calculated for each scan line. The threshold calculated becomes the threshold for the next scan line. The processed analog data are compared to their respective, either a binary edge or a binary hot or cold, threshold. If the data exceed their threshold, then a logical threshold is generated. It will remain one as long as the data exceed the threshold.

In Figure 25, the edge signal derivation is shown. It is two-dimensional in that three horizontal scan lines are used in calculating the edge. The implementation shown is for the horizontal component only. Three lines of video are added together. A delayed version of this sum is then subtracted to give a positive voltage for a rising edge and a negative voltage for a falling edge. The absolute value is then taken, giving all positive voltage values. The data are compared to the threshold T_E. When edge is greater than T_E, the comparator then produces a logic one signal.

The edge threshold is calculated as shown in Figure 26. The edge signal is integrated over the line time to get the average edge signal during one horizontal TV line. During retrace, the input to the integrator is grounded. The average output is recursively filtered using the parameter "a" as the parameter that determines how many previous values are used. The previous sampled and held value is multiplied by "a" and the edge average is multiplied by (1-a). This value is sampled during retrace and multiplied by the constant K_E which gives the threshold value T_E.

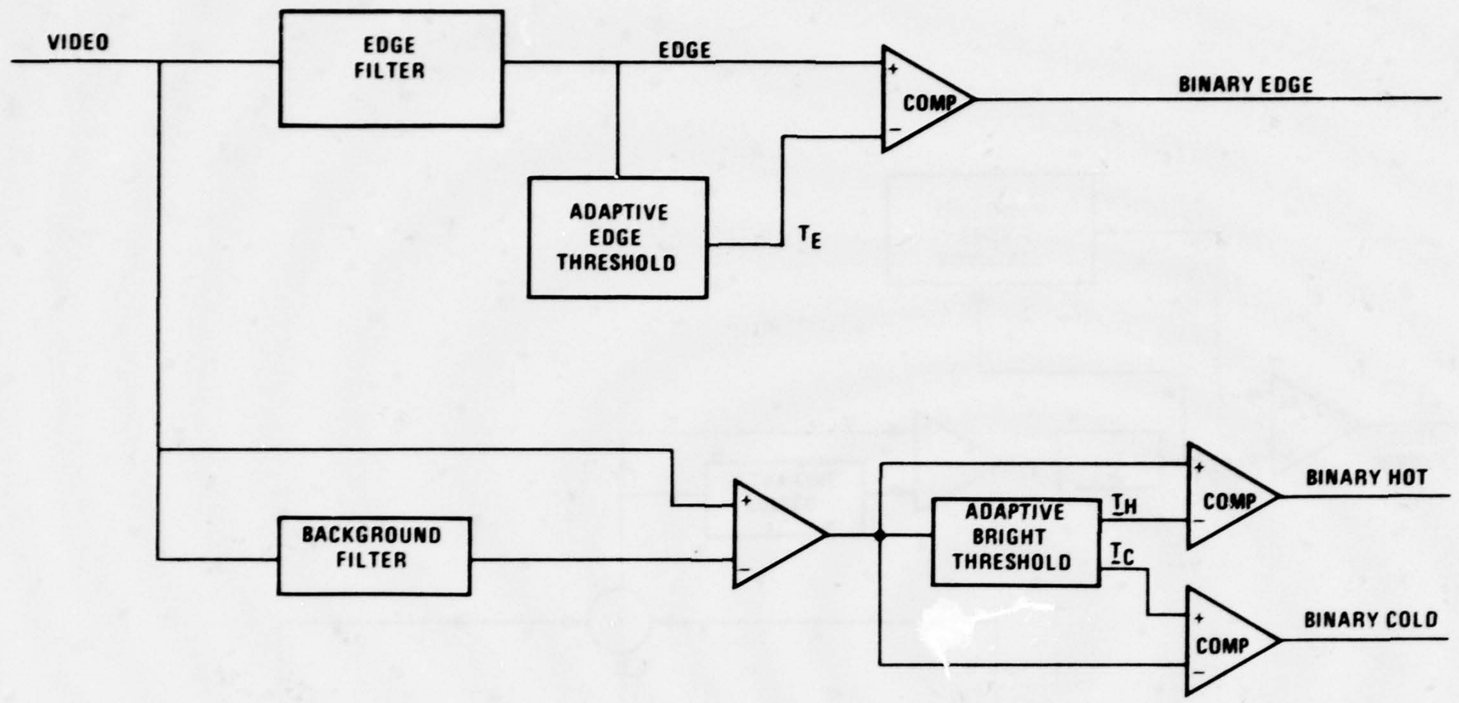


Figure 24. Autothreshold Structure

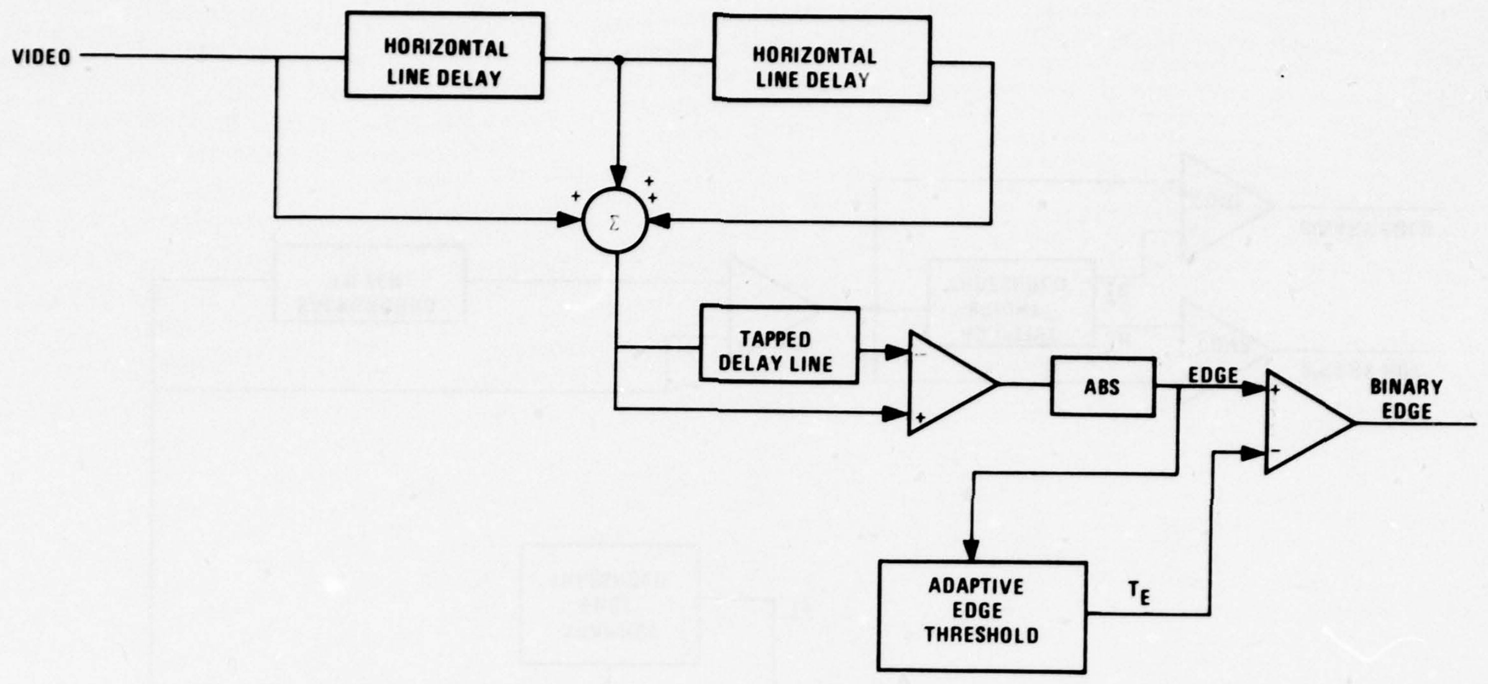
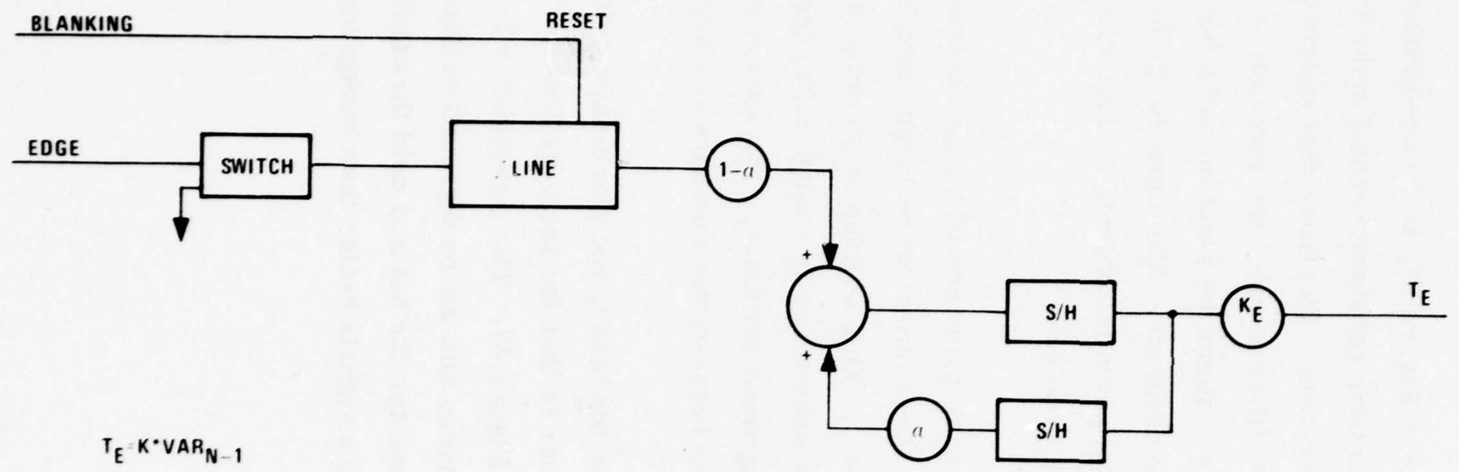


Figure 25. Edge Signal Derivation

ADAPTIVE EDGE THRESHOLD

ADAPTIVE EDGE THRESHOLD



$T_E = K \cdot VAR_{N-1}$

$VAR_N = \alpha VAR_{N-1} + (1-\alpha)\frac{1}{N}\ e$

Figure 26. Adaptive Edge Threshold

The background filter, shown in Figure 27, is a two-dimensional recursive filter which operates with two time constants--one for the vertical direction and one for the horizontal direction. The filter that operates in the horizontal direction is a low pass RC filter whose time constant is a few pixels. The vertical filter consists of a summing junction and a horizontal line delay for storage of the background estimate. The input to the background estimate is zero during times of a potential object, as indicated by the presence of an interval on the previous scan line.

The switch is needed because the presence of a high contrast object would get into the background, and if the object were large enough, the background estimate might equal the object. This buildup is shown graphically in Figure 28. The switch is turned off when there is enough difference between the object intensity and the background estimate. The switch is turned back on when there is little difference between the intensity and background estimate.

The bright threshold used for the binary hot and binary cold derivations is calculated in a similar manner to that for the edge filter. The calculation implementation is shown in Figure 29. The threshold is based upon the difference between the raw video and the background estimate. There are, however, separate multipliers for the hot and cold threshold, so that different thresholds can be used for signals hotter than background and colder than background.

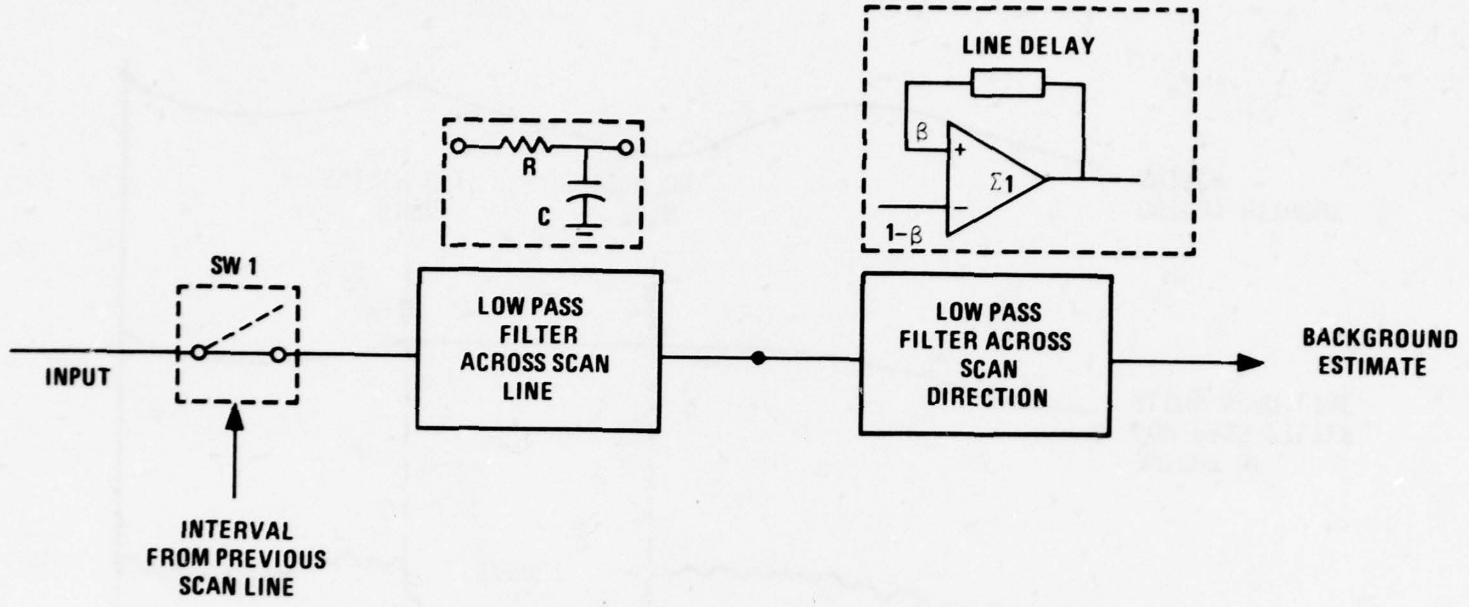

SW 1
INPUT
INTERVAL FROM PREVIOUS SCAN LINE
R
C
LOW PASS FILTER ACROSS SCAN LINE
LINE DELAY
β
+
$\Sigma 1$
−
$1-\beta$
LOW PASS FILTER ACROSS SCAN DIRECTION
BACKGROUND ESTIMATE

Figure 27. Background Estimator

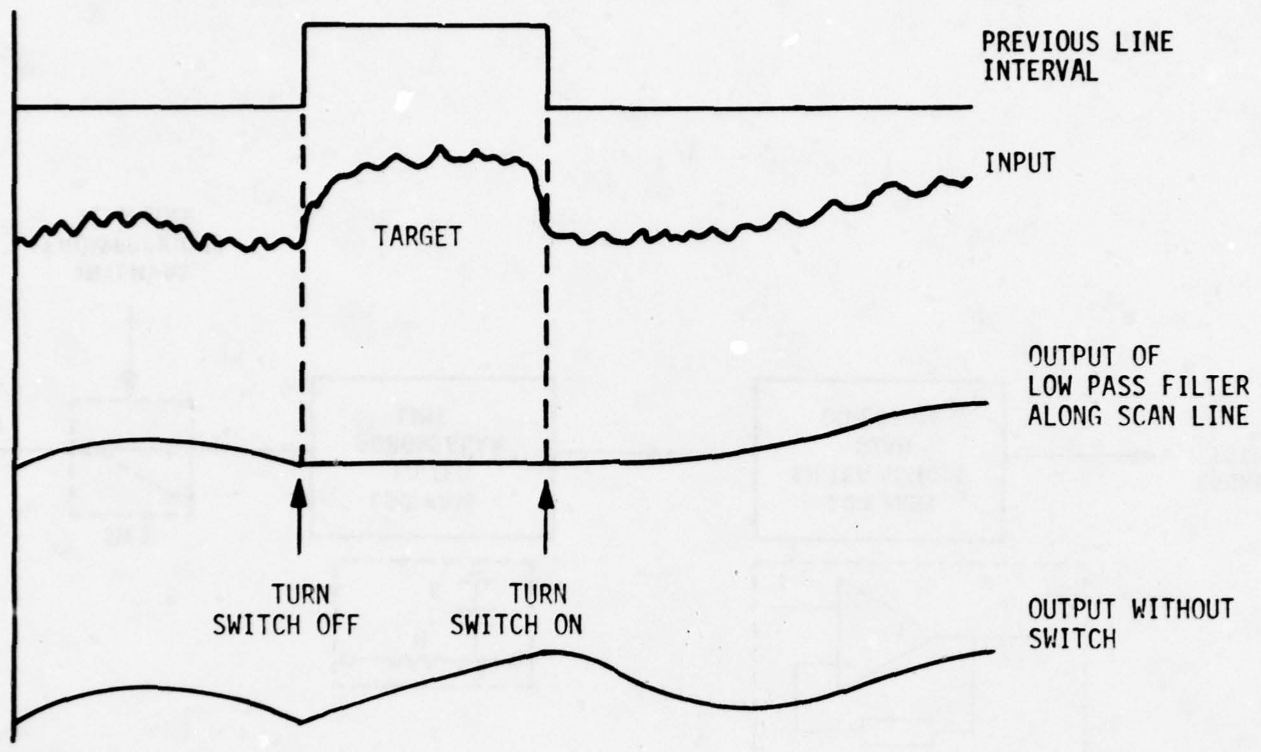


Figure 28. Background Filter and Switch Rationale

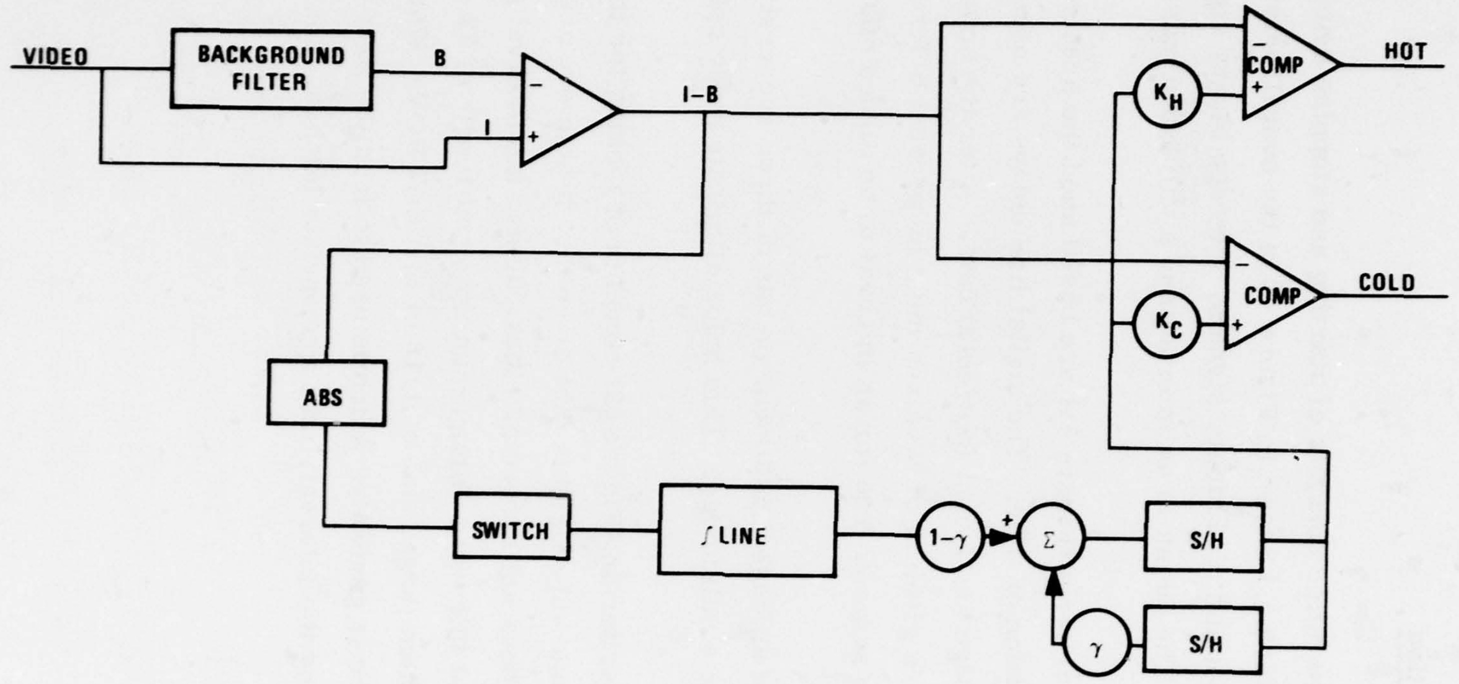


Figure 29. Adaptive Bright Threshold

Interval Generation

The interval generation consists of starting and stopping criteria as well as a validation criterion. Shown in Figure 30 is the overall interval generation. Inputs are the hot and cold binary signals, the edge binary signal, and a system clock. The output is an interval and a validation pulse.

The start/stop criteria (Figure 31) are based upon the presence of a signal for a given duration of time. The digital line delays are used to get proper phasing of the edge and bright (hot/cold) data. After the appropriate delays are introduced, a given M of N values must be present before an interval is turned on. The same is true for an interval to be turned off.

The presence of an edge must occur on one of three successive scan lines in order to have a valid edge. This helps eliminate noise spikes.

The validation criterion (Figure 32) consists of looking for the coincidence of valid edges and valid brights (hot or cold). If both occur and the length of the interval does not exceed 32 clock pulses, the interval generated is declared a valid interval. A maximum interval length of 32 clock pulses allows a maximum target size of 1/16 of the total FOV. When the interval is valid, a pulse is generated which is used to load the first level features, generated during the interval, into appropriate latches.

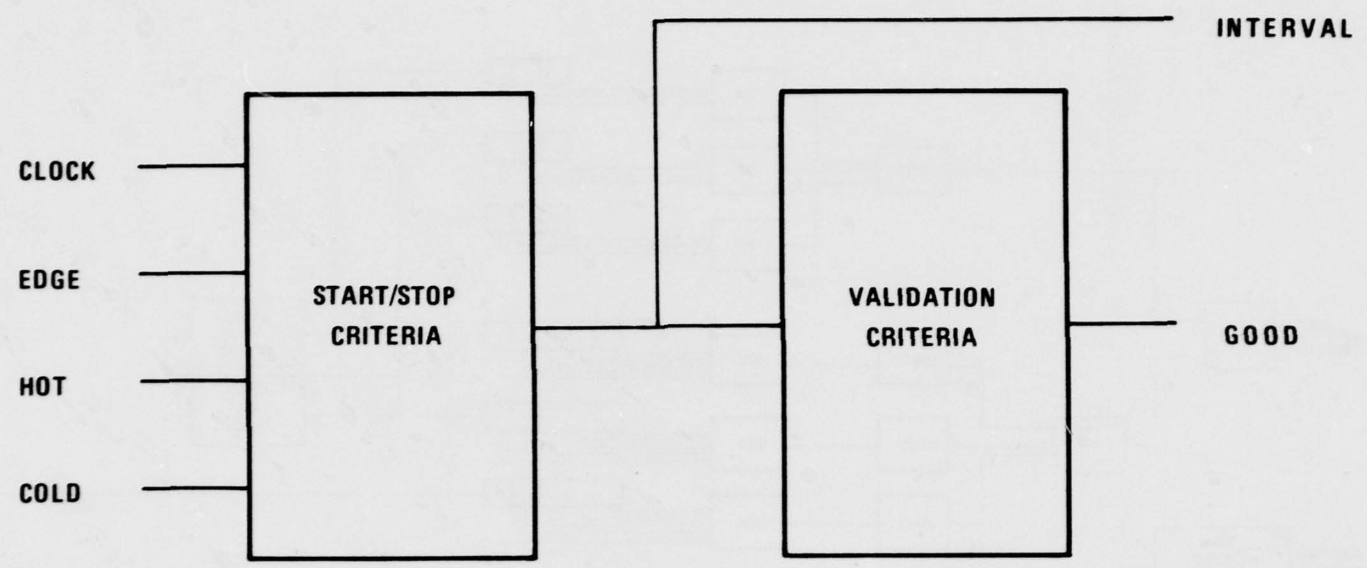


Figure 30. Interval Generation

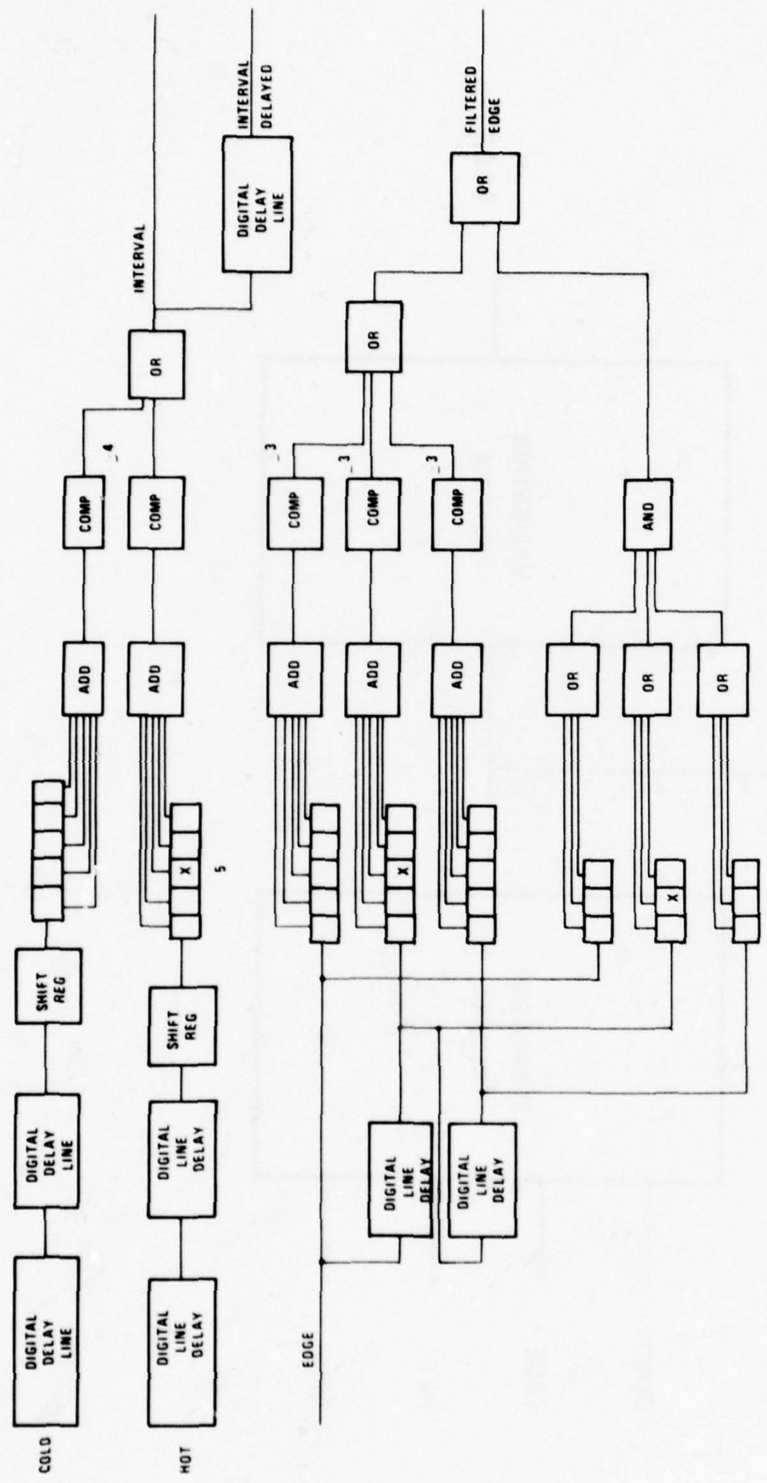


Figure 31. Start/Stop Criteria

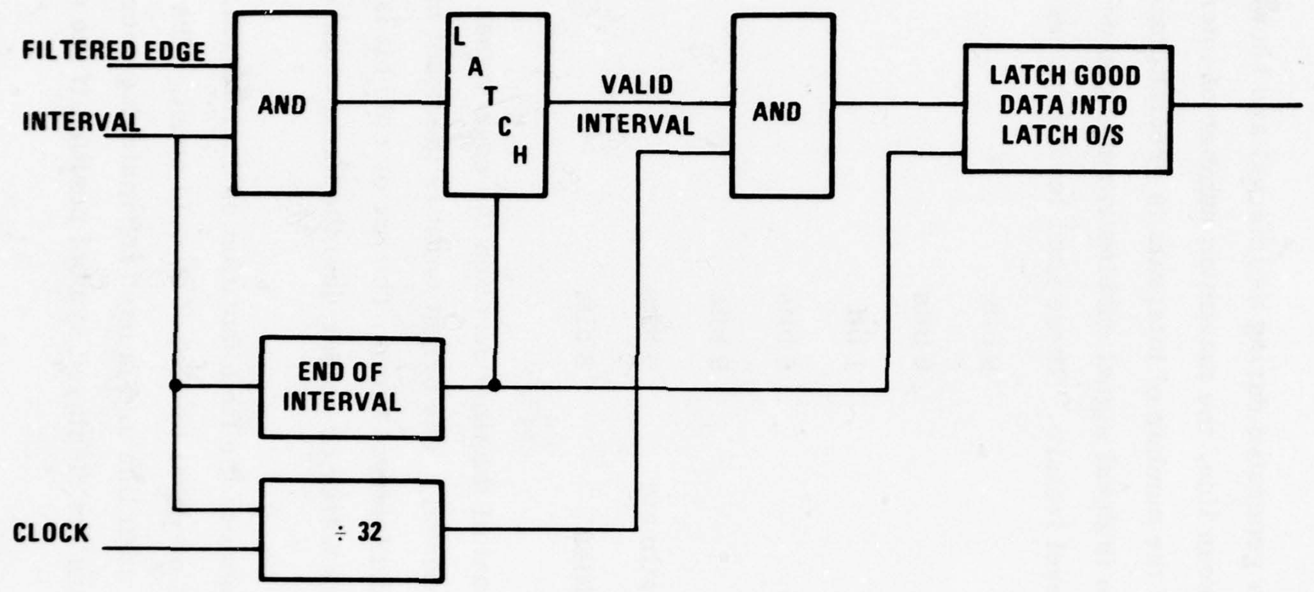


Figure 32. Validation Criteria

First Level Features

The first level features generated during an interval are shown in Figure 33. During one horizontal scan line, the maximum number of intervals that will be accepted are 16. If the number of intervals is greater than 16, an overflow is generated. The interval signal enables various counters whose output value is the first level feature. These first level features are:

• Edge count	6 bits
• Bright count	6 bits
• Hot or cold	1 bit
• Interval width	6 bits
• X position	9 bits
• Background estimate	8 bits
• Sum of intensities	8 bits

The edge count is a count of the time duration the edge signal is above the threshold during the interval. The bright count is the count of either the hot or cold signal during the interval time. The hot or cold bit is determined by whether the object is hotter or colder than the background estimate.

The width count is a count of the time duration of the interval. It should always be greater than or equal to the hot or cold count. The X position is the position within the scan line and is used for matching successive intervals and for determining the displayed symbol position if the object is a target.

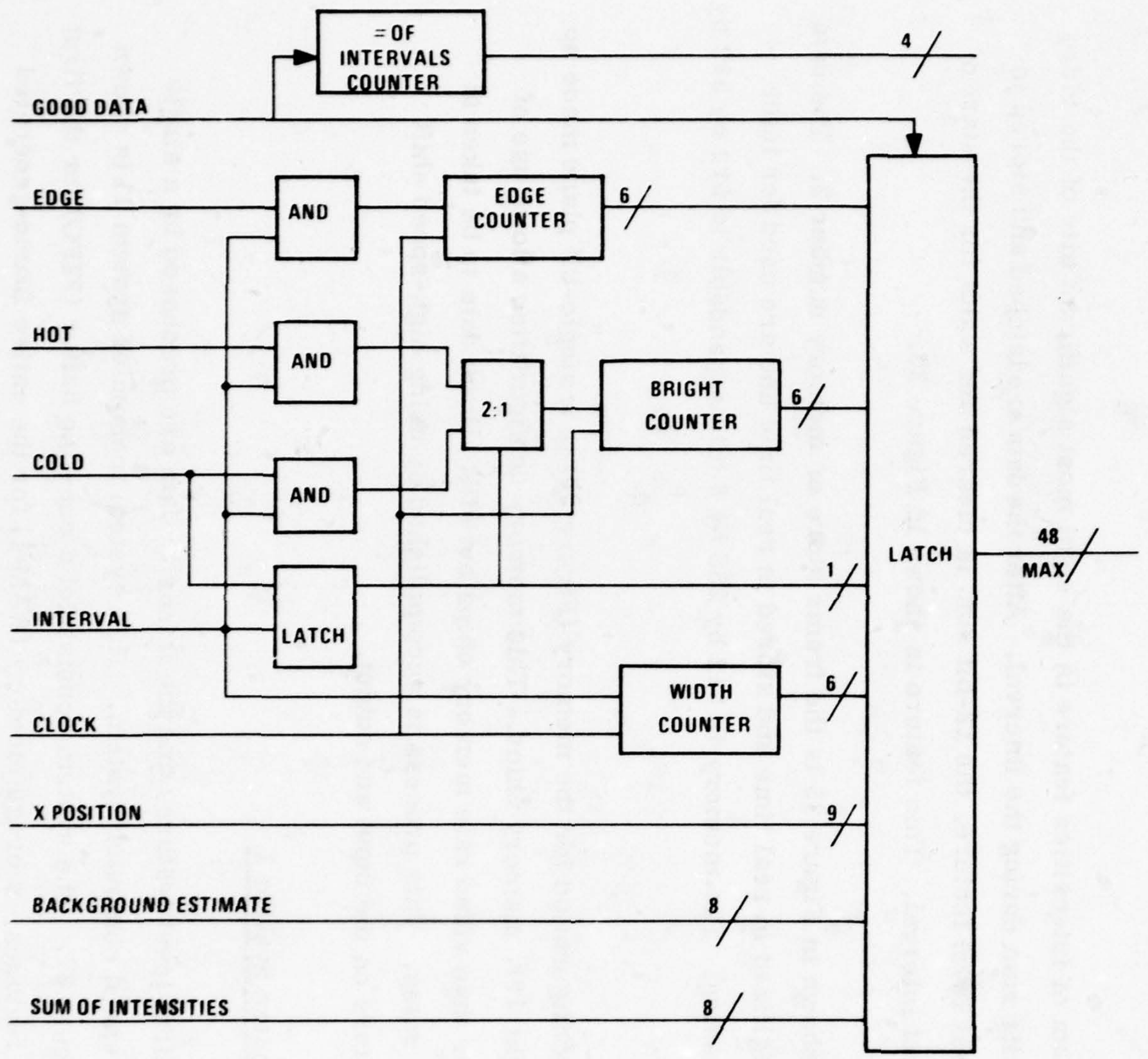


Figure 33. First Level Features

The background estimate (Figure 34) is a sample of the analog background estimate near the start of an interval. It is digitized and then latched as a first level feature.

The sum of intensities feature is the eight most significant bits of the video intensity sum during the interval. After the data are latched and stored in the first level feature, the 12-bit sum is cleared and waits for the start of the next interval. This feature is shown in Figure 35.

Also shown in Figure 35 is the frame store or memory number 2. The data are digitized in real time and stored in real time and are used for later processing. The memory is 512 by 256 by 6 bits expandable to 512 by 512 by 8 bits.

The configuration for the memory (Figure 36) is a single-bit plane made up of eight 16K memory chips. This memory configuration allows use of slower than video rate memory chips but still allows data to be taken at video rates. This process is accomplished by using high-speed shift registers on the input and output.

Computer System 1

The first level features and the intensity data are processed by a single high-speed computer system. This system (computer system 1) is shown in Figure 37. The system consists of a one-line buffer (FIFO) for the first level features; storage memory (RAM 1) for the entire frame-generated first level features and scratch pad area; a 2901-based CPU; and a hardware

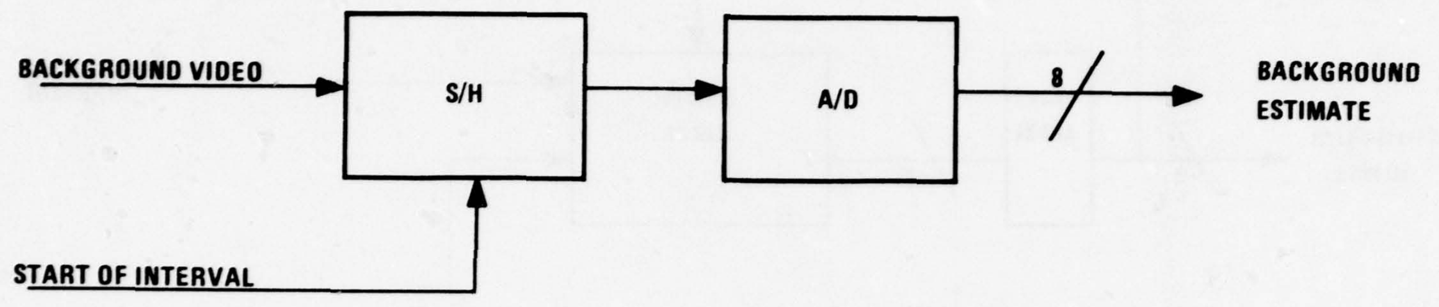


Figure 34. Background Feature

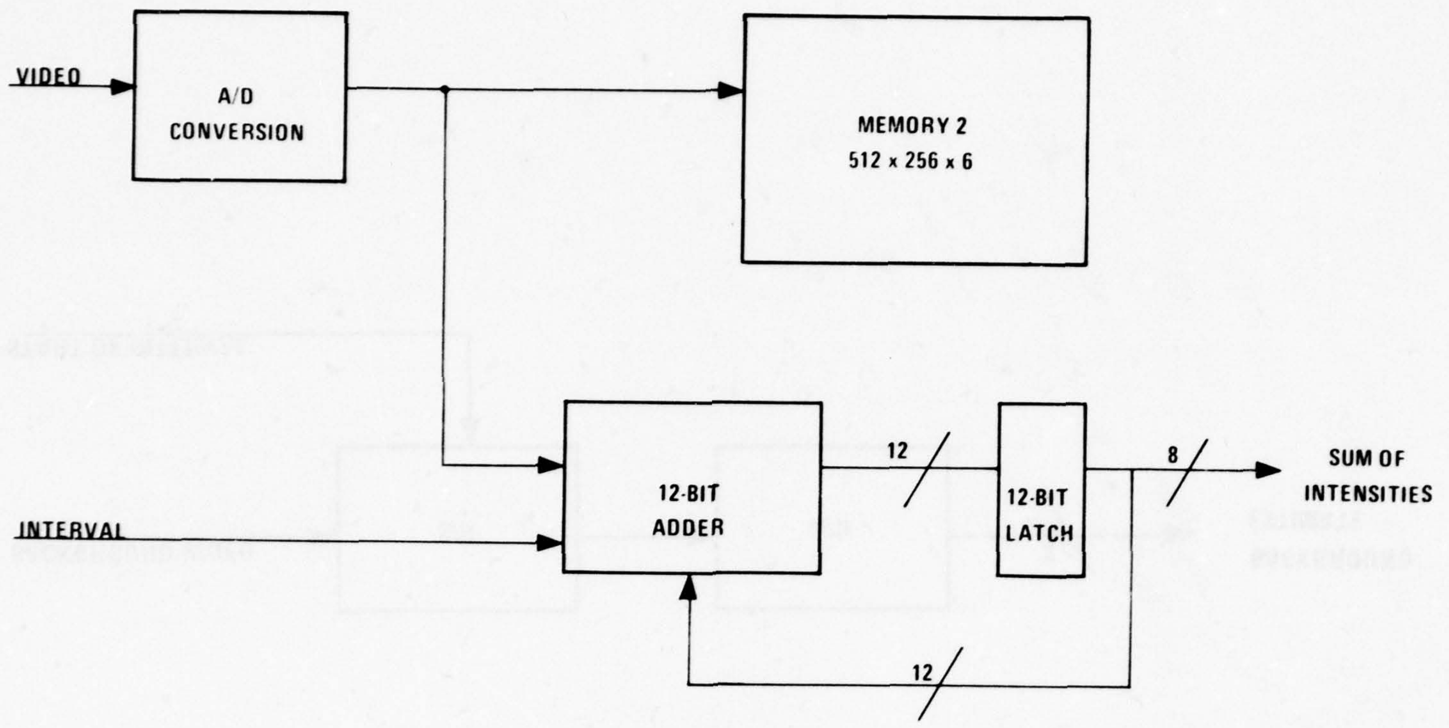


Figure 35. Intensity Feature

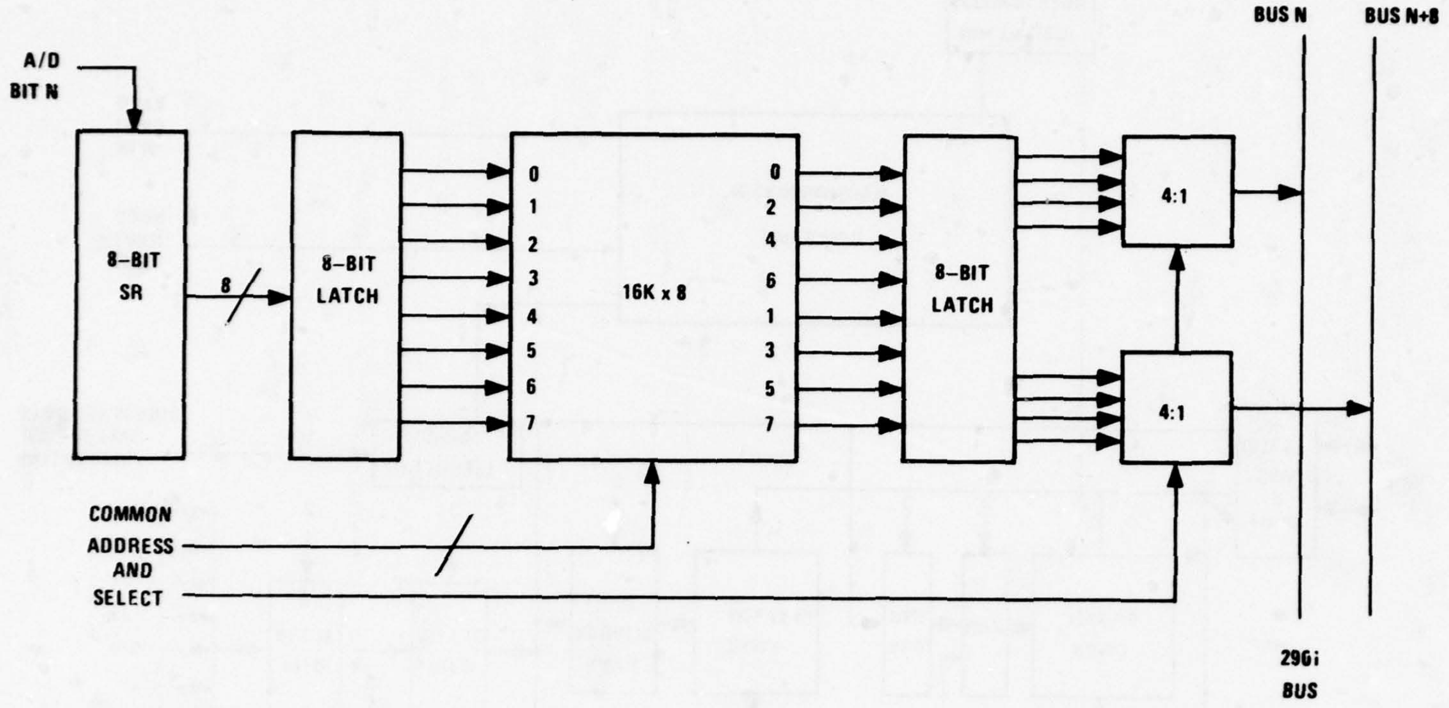


Figure 36. Video Field Memory Bit Plane

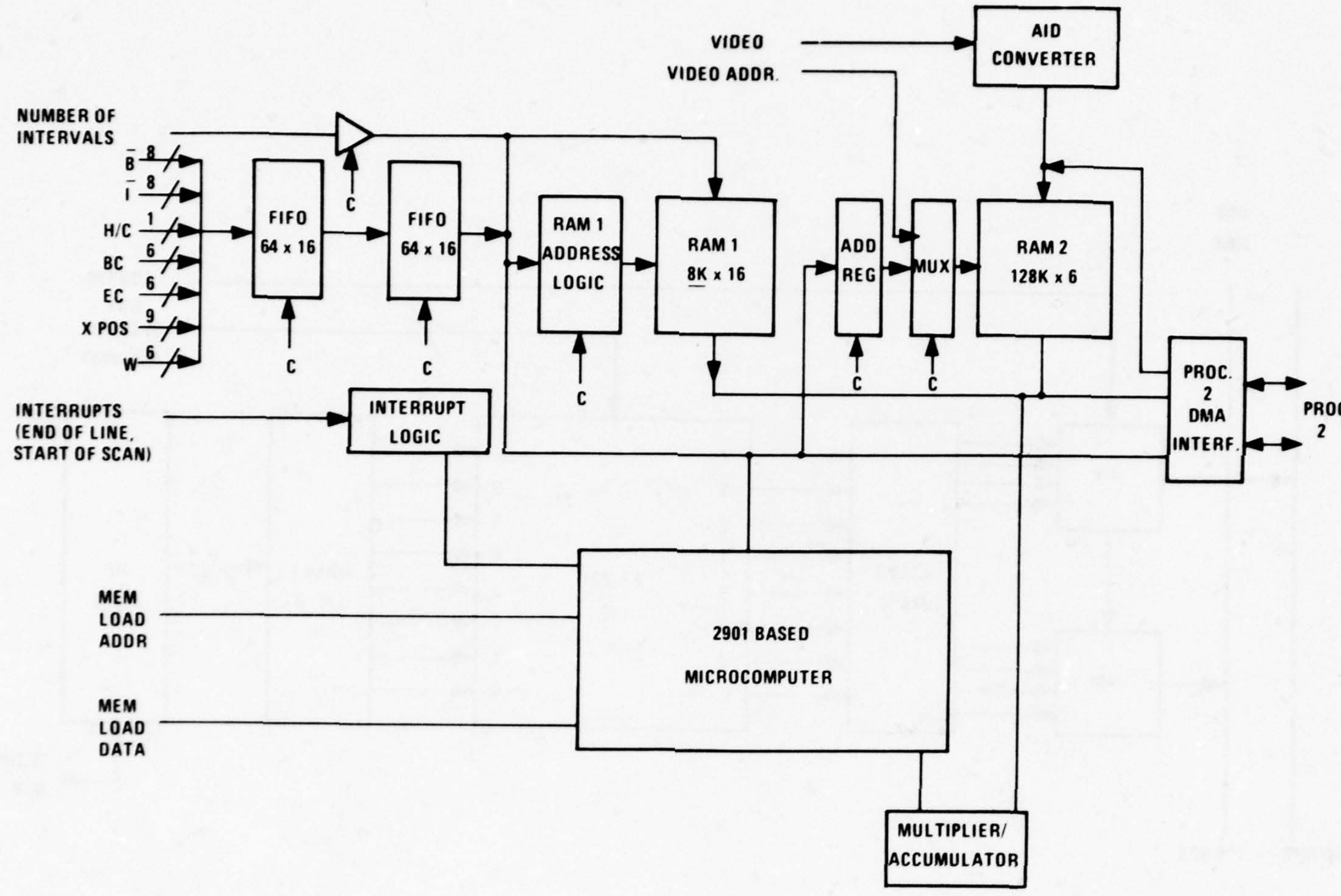


Figure 37. Computer System 1

multiplier-accumulator. RAM 2, already discussed, can also be considered part of the computer system. A direct-memory access interface allows for reading and/or writing into RAM 1 or RAM 2 by a second CPU.

The RAM 1 or Memory 1 configuration is shown in Figure 38. Both FIFO memories are configured 64-words long by 16-bits. Since there are 48 bits of first level features, the data must be multiplexed before going into FIFO 1. Then the data are direct-memory accessed into RAM 1. At the end of the scan line all the data in FIFO 1 are transferred to FIFO 2. The transfer of data into RAM 1 may take longer than the horizontal retrace time if the number of intervals is quite large. However, by doing a fast transfer from FIFO 1 to FIFO 2, an entire scan line of time is available for transfer.

Computer System 2

After the frame data are processed, target position and class data are transferred to computer system 2 (Figure 39). During operation, the data are analyzed and a symbol of the proper size and type is generated. The joystick, CRT, and floppy disk are used during the diagnostic checkout and training phases. Target locations are identified for training by the joystick, and the statistical data for various classes are stored on the disk.

Symbol Generation

The symbol generation consists of writing to a one-bit plane of memory with location synchronized with the video. The symbol generation scheme is shown in Figure 40. The symbol to be generated is controlled by software and hence can be easily changed by changing a few locations in computer system 2 program memory.

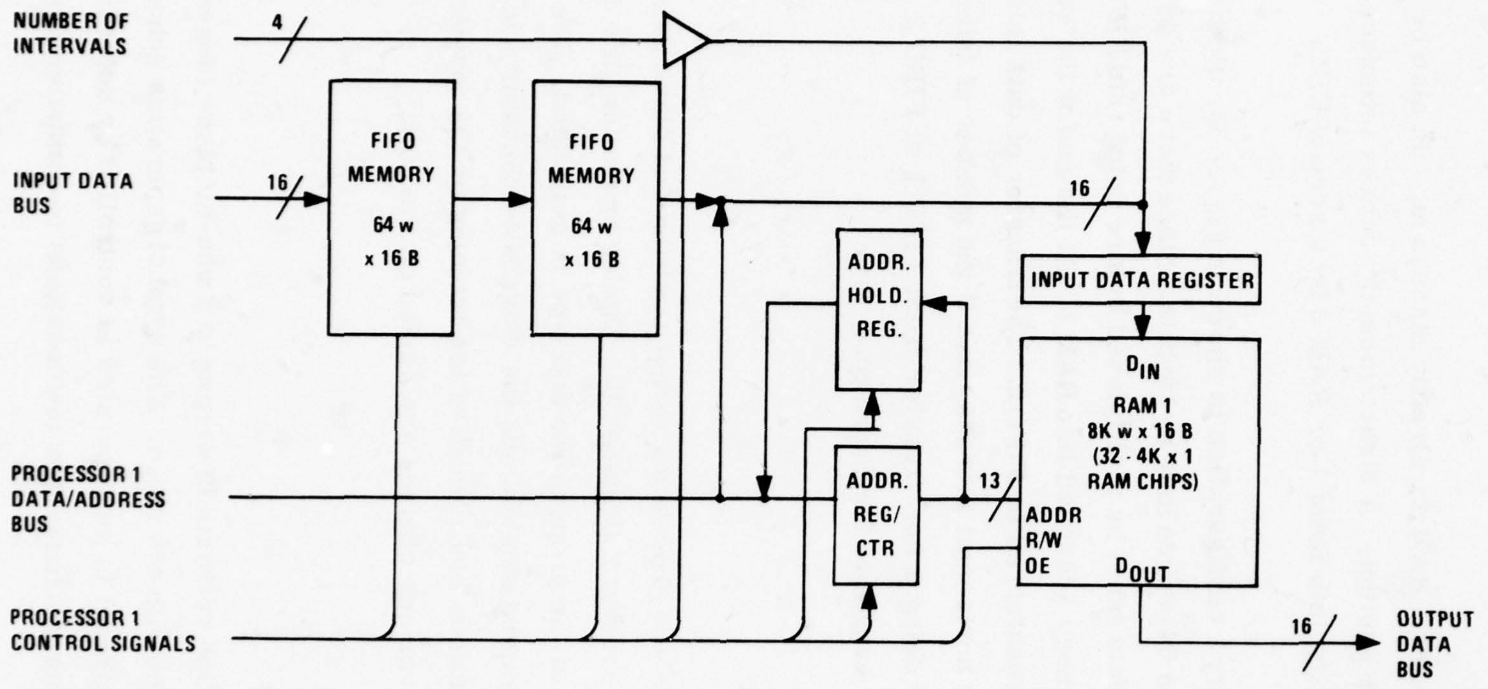


Figure 38. Memory 1 Configuration/Interface

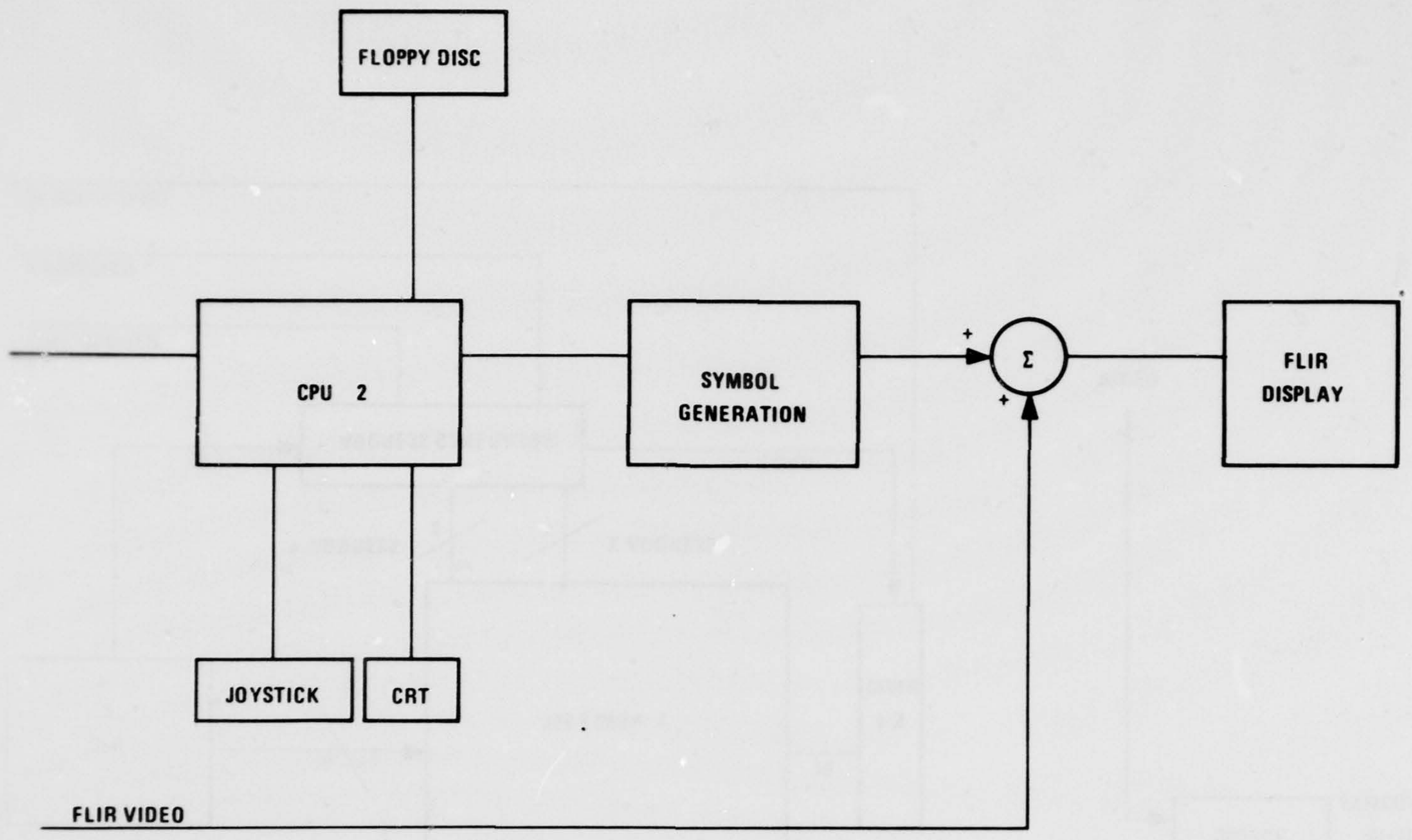


Figure 39. CPU 2 Configuration

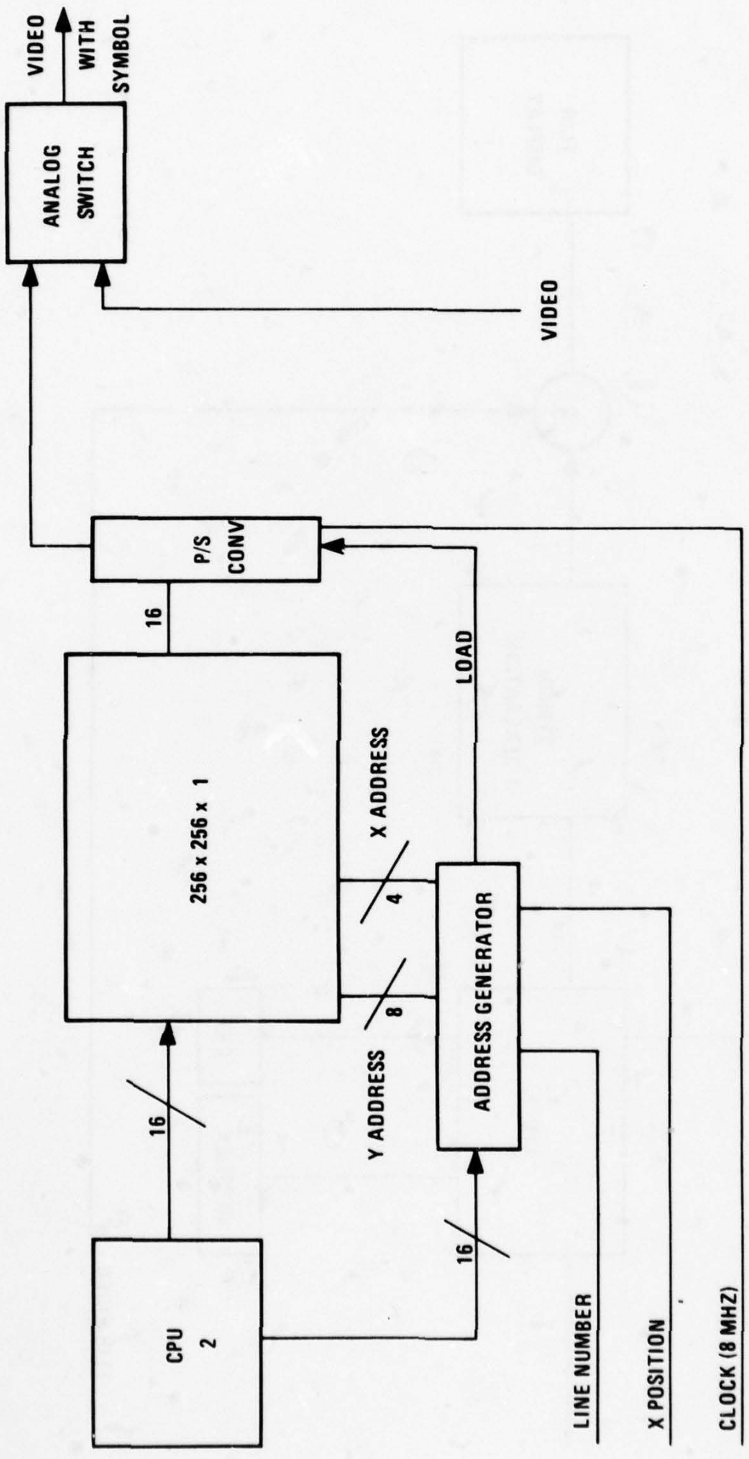


Figure 40. Symbol Generation

TIMING AND SYNCHRONIZATION

Associated with the operation of PATS is a master timing and synchronization scheme (see Figure 41). The system must be capable of working with either 525- or 875-line video data. The number of samples/line remains fixed at 910 samples/line. The digital data will be sampled at 512 samples per horizontal line. Also included will be standard TV signal such as composite blanking, vertical blanking, and horizontal blanking. A separate clock will be provided for the CPU 1 cycle time.

PHYSICAL CONFIGURATION

It is anticipated that all the PATS hardware including both CPUs will fit into one ATR (air transport racking) large chassis. Some of the external functions to be provided will be test points, power on, enhancement on/off, and target priority mode switch. Conceptually, this configuration will look like Figure 42. The video level indicates the presence of video at the input to the target screening function. Symbol control will also be provided so that only a given class will be displayed.

Internal to the chassis will be space for 28 wired cards, though this may decrease because some will be wire wrap cards and some will be printed circuit. The approximate layout is shown in Figure 43. Power supplies and cooling will be an integral part of the PATS chassis.

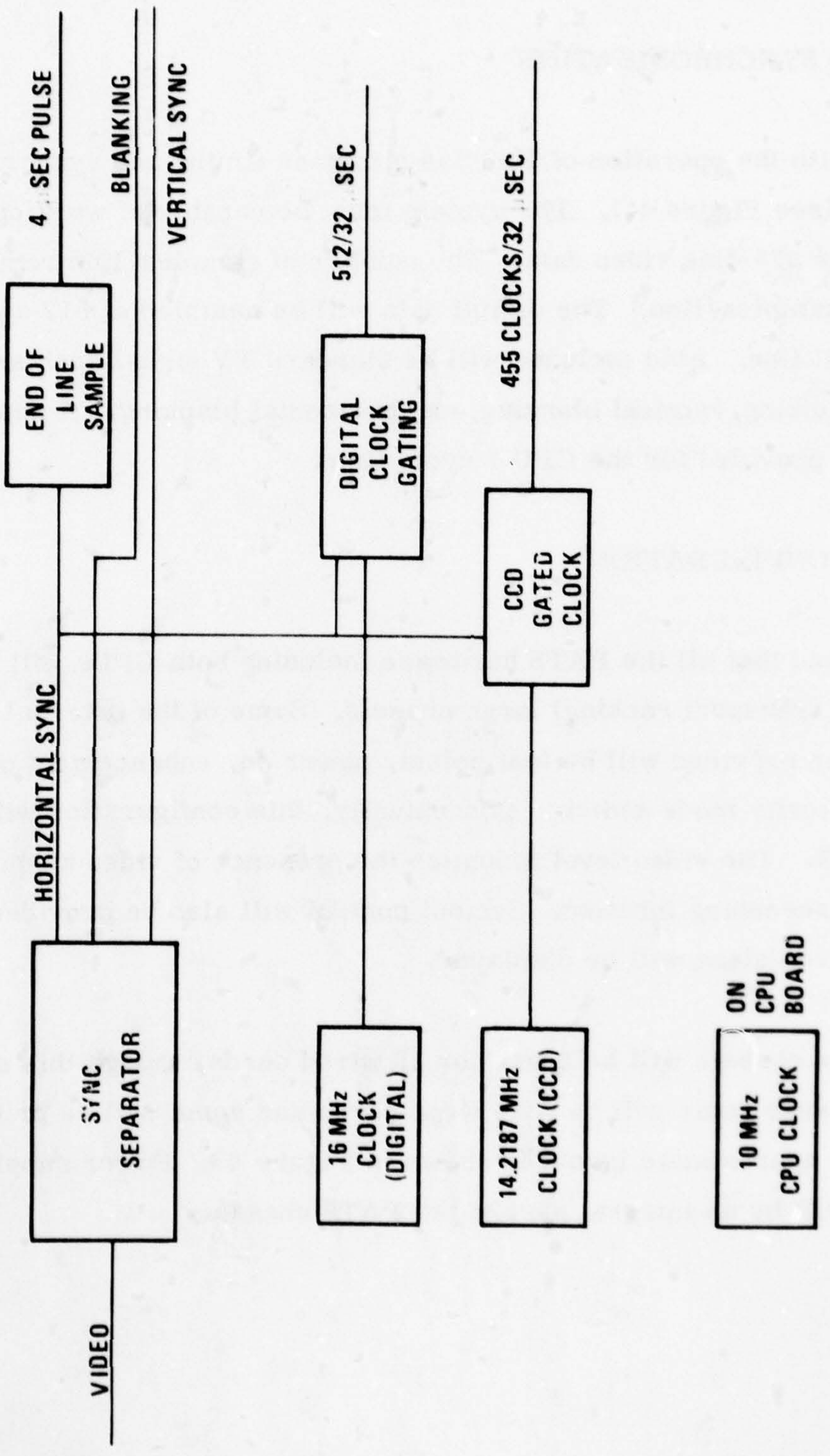


Figure 41. Timing and Sync

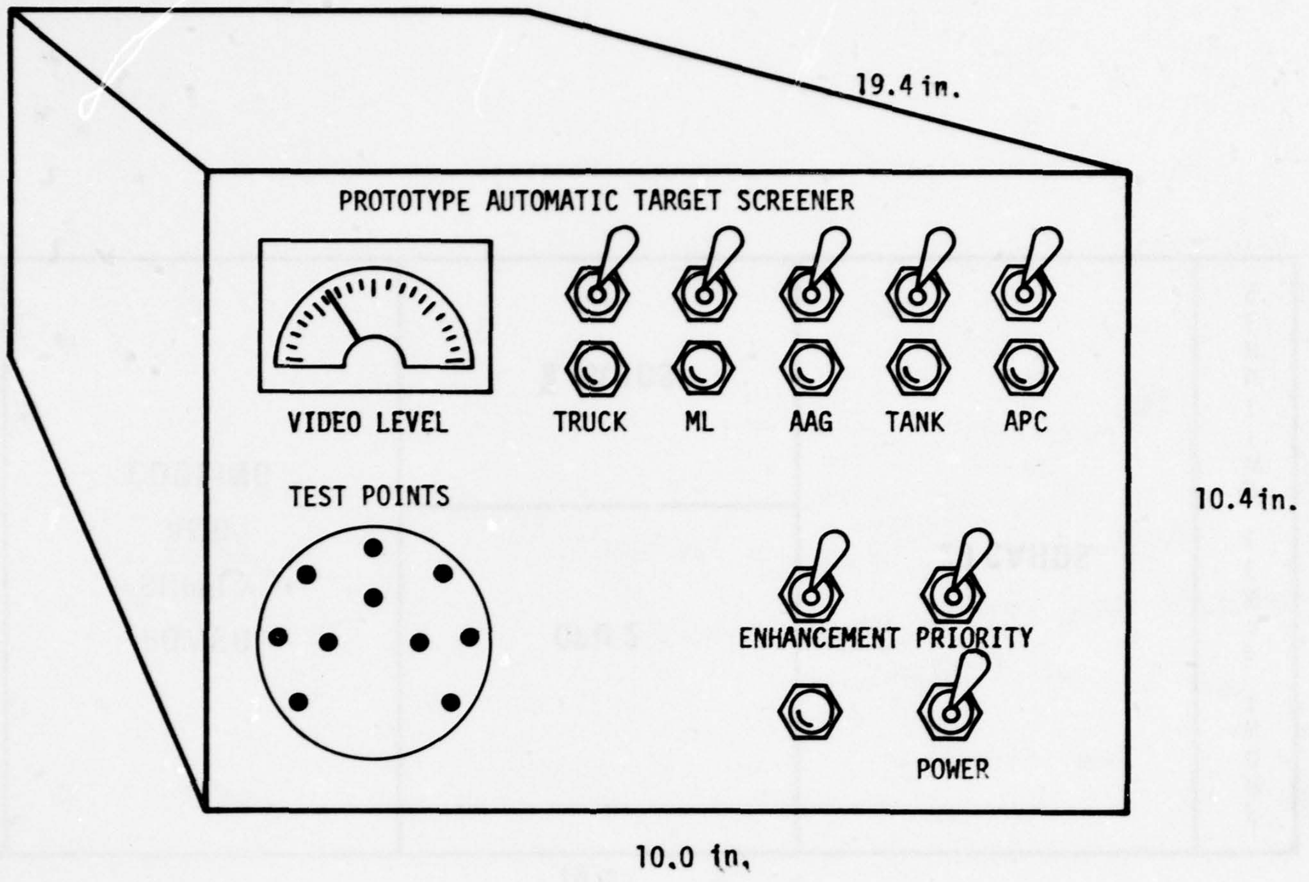


Figure 42. PATS Physical Configuration

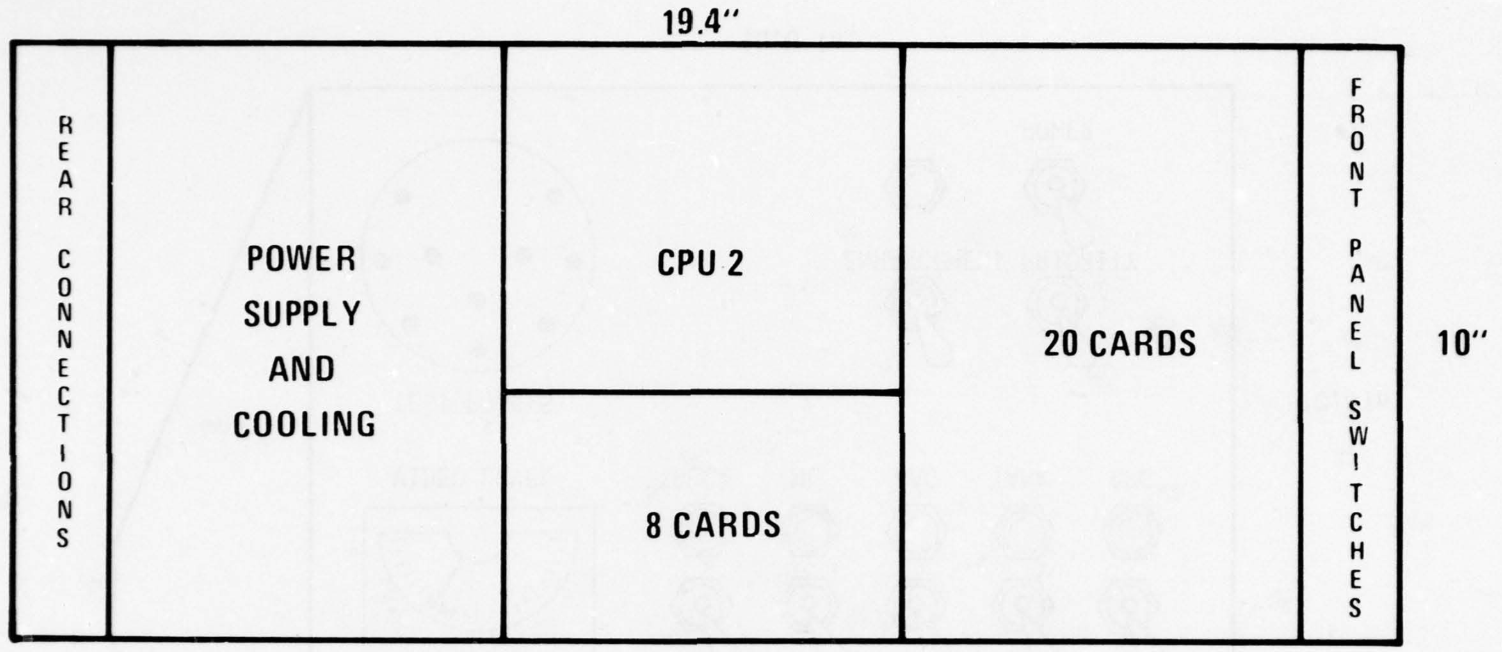


Figure 43. PATS Physical Layout

SOFTWARE DESIGN

Work has begun on designing the implementation software for PATS. The algorithms that are used in the simulation are being put into a form which will run within the system architecture proposed in the previous subsection.

Figure 44 diagrams the proposed software functions. As was discussed in the hardware subsection, these functions are partitioned between two CPUs. CPU 1 segments a frame of video data into objects, classifies these objects, and then reports classification, size, and location to CPU 2. CPU 1 processes data at the rate of ten frames per second; that is, all processing on a single frame must be completed within 0.1 second. CPU 2 accumulates data passed to it from CPU 1 over several frames, and based on an analysis of these interframe data, generates a graphics overlay on the operator's video monitor which highlights and identifies detected targets. In addition, CPU 2 performs nonreal-time diagnostics on CPU 1 hardware and software. The software functions of both of these CPUs is discussed in more detail below.

Data Acquisition

CPU 1 receives image data via the interval and intensity memories shown in Figure 44. These memories buffer real-time data so that they can be processed by CPU 1 on a nonreal-time basis. The autothreshold hardware processes every sixth video field and dumps real-time interval data to the interval memory via DMA transfer on a scan line by scan line basis. These data are processed by CPU 1, which then stores the processed data back into the interval memory for later use. It should be noted that this fast (under 200 ns access) 16K by 16-bit memory also serves as a scratch pad

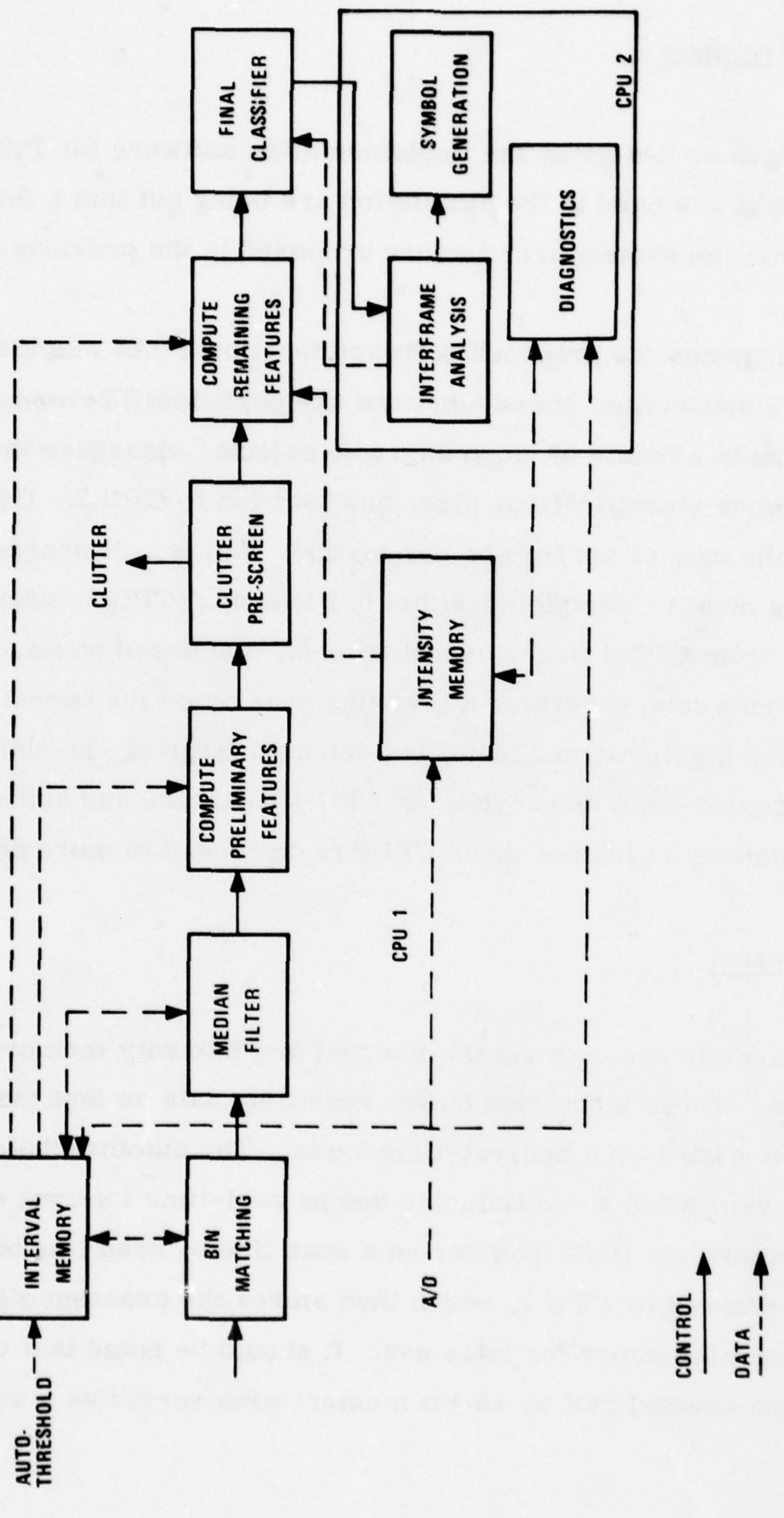


Figure 44. Software Functions

for CPU 1. The same frame that is processed by autothreshold is also digitized and stored in the intensity memory. This memory is accessed by CPU 1 to obtain intensity information whenever it computes intensity moments on an object.

Interval data produced by autothreshold during a scan line are dumped to the interval memory during horizontal retrace in the format shown in Figure 45. The block of data corresponding to a given line is headed by a packed word giving the line number and the number of intervals which are generated for that line. Following this word is a group of 3 x (number of intervals) words containing the packed features from each interval; i.e., three words are used for each interval. Currently, the features packed into these words are interval color (i.e., hot or cold--H/C), starting X position, interval width, the sum of intensities over the interval (ΣI), background filter sample ($\overline{B}$), and interval bright and edge counts. No data are placed in the interval memory for scan lines during which no intervals were generated. A register maintained by the hardware indicates how much data have been loaded into the memory.

Data Processing

Bin Matching--The task of the bin matching procedure is to associate the one-dimensional intervals characterized by the interval features into sets of intervals which determine two-dimensional objects; that is, CPU 1 essentially must read interval data from the memory, reorder them, and then write them back. The bin matching algorithm has been discussed in a previous report[1] and will not be described here in detail. Figure 46 lists most of its key points. The algorithm is sequential, so that at any time

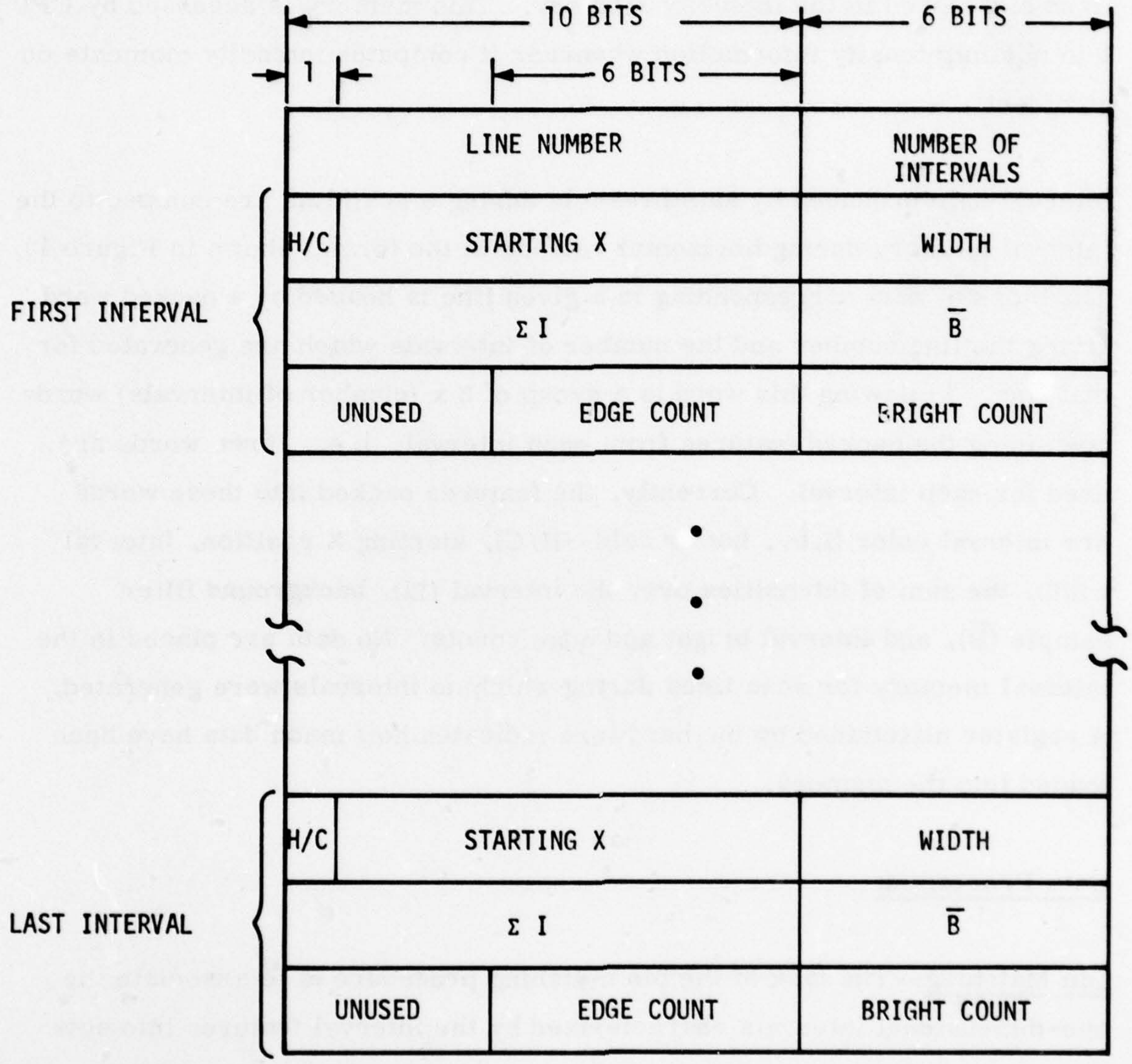
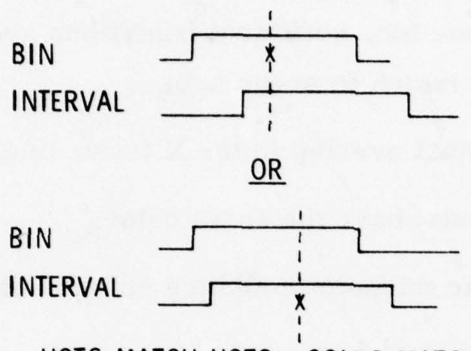


Figure 45. Interval Data Format

- GROUPS INTERVALS INTO OBJECTS
- OPERATES ON PAIRS OF INTERVALS
- INVOLVES MIDPOINT COMPUTATION
- SYMMETRIC MATCHING CRITERIA



- HOTS MATCH HOTS; COLDS MATCH COLDS
- OBJECTS WITH LESS THAN 2 INTERVALS ARE THROWN OUT
- 2901 BASED

Figure 46. Bin Matching Software

during processing of a frame, a number of partially completed objects or "bins" will exist. Very simply, each bin accumulates object intervals on successive scan lines that fall within the midpoint of the previous interval assigned to the bin. Missed intervals are filled in up to three lines. If there are no new intervals or scan lines, the bin is considered closed and further processing is then done on the bin (such as computing the various elementary features).

Work is proceeding to develop an optimized version of the bin matching algorithm, which when implemented, will minimize the expected number of comparisons needed to match an interval against a list of active bins. Bins will be sorted in ascending order according to the value of the midpoint of the last interval associated with each bin. In this way, the relative locations of bins are established across an image. Given an interval to be compared against this ordered list of bins, the algorithm will then try to establish the necessary conditions to achieve a bin match. Once these necessary conditions have been satisfied by some bin, sufficient conditions are checked. The necessary conditions for a match to occur are:

1. The bin and interval must overlap in the X (scan line) direction.

2. The bin and interval must have the same color.

The sufficient conditions are the midpoint matching criteria shown in Figure 46.

Listing the bins in ascending order will make establishment of the first necessary condition above very inexpensive and will make determination of when to stop comparing bins possible (that is, once a bin is found entirely to the right of an interval, it will be impossible to obtain a match with that bin and all bins following it). Comparisons to establish the first necessary condition will start with the bin nearest the left edge of the image and continue with those increasingly toward the right. Bins which do not satisfy the necessary conditions for matching an interval will each require at most three comparisons with the interval; most will require only one. Establishing the sufficient conditions will require four comparisons. Therefore, a bin that matches an interval will require seven comparisons using this scheme. However, most bins will not even overlap the interval and therefore will require only one comparison.

Figure 47 shows a map of the way the interval memory during bin matching is expected to appear. Interval features are loaded sequentially in real time into the upper 8K words of the interval memory by the autothreshold

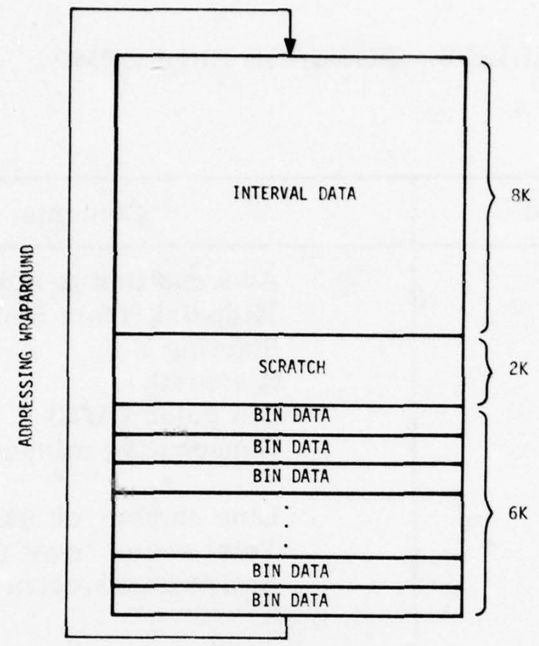


Figure 47. Map of Interval Memory

hardware, using the data format in Figure 45. Two thousand words are reserved for scratch storage required by the bin matching procedure. The bins are developed in the 6K memory words following this scratch area with wrap-around into the upper 8K. Note that bin matching essentially copies interval data into the bins, so redundant interval data in the upper 8K of memory can be overwritten. It is expected that a fixed length block of approximately 160 words will be allocated for each bin; this block will contain bookkeeping data necessary while a bin is open (e.g., bin endpoints, midpoint, missed scans count, etc.) and features computed over the entire bin after it is closed (e.g., total brights, moments, etc.). Table 8 shows a possible format for this bin data. Active or open bins are linked together in ascending order using a linked list structure; once a bin is closed, it is removed from this list. Interval features that are unpacked to do bin

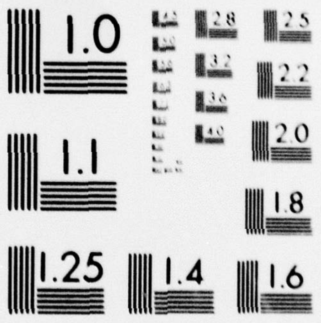
TABLE 8. POSSIBLE BIN FORMAT

	Word	Contents
Bookkeeping	1	Address link to next bin
	2	Midpoint (from last interval in bin)
	3	Starting X
	4	X + width
	5	Bin color (H/C)
	6	Consecutive missed scans count
	7	Line number on which bin starts
	8	Total scans count (N)
	9	Active scans count
First interval in bin	10	X
	11	Width
	12, 13	Packed features
	⋮	⋮
Last interval in bin	4N + 6	X
	4N + 7	Width
	4N + 8, 4N + 9	Packed features
	4N + 10 ⋮ 160	Object features (target contrast, moments, etc.)

AD-A060 849   HONEYWELL INC MINNEAPOLIS MINN SYSTEMS AND RESEARCH --ETC F/G 17/5
PROTOTYPE AUTOMATIC TARGET SCREENER.(U)
JUN 78 D E SOLAND, P M NARENDRA, R C FITCH   DAAK70-77-C-0248
UNCLASSIFIED   78SRC54-2   NL

2 OF 2
AD
A060 849

END
DATE
FILMED
1-79
DDC

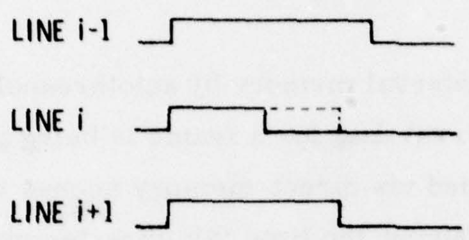

1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

matching (e.g., starting X and width) are stored unpacked in the bins to avoid the overhead of having to unpack them again later. Features which are not needed for bin matching are left in their packed form. Note that the starting X and width of each interval within a bin must be saved for the median filter and for moment feature computations.

Bin matching and loading of the interval memory by autothreshold will proceed in parallel while the interval data for a frame is being generated. The interval memory will be loaded via direct-memory access transfers from buffers in autothreshold. During the time this direct-memory access is taking place, CPU 1 will not be able to access the memory. However, between these transfers CPU 1 will try to do as much processing as it can. If the frame being processed does not produce many intervals, it is possible that bin matching will be complete by the end of the frame. If bin matching is not completed, additional time will be used to finish it.

Median Filtering--Once bin is complete, CPU 1 will smooth the boundaries of each bin using a one-dimensional median filter of width three. The values which are input to the filter are the endpoints of the intervals making up each bin. A separate filtering operation is done on the left- and right-hand edges of each object (see Figure 48). Each triplet of endpoints on a given edge is sorted, and the middle endpoint of the triplet is assigned the middle value from the sort. Note, however, that this filter is nonrecursive; only original endpoints are input to the filter, not filtered ones. CPU 1 will process all the intervals within each bin and update, where necessary, starting X values and widths.

- SMOOTHS BOUNDARIES OF OBJECTS FOUND BY BIN MATCHING
- OPERATOR ON INTERVAL TRIPLETS
- INVOLVES SORT OF 3 VALUES (2 COMPARISONS) ON EACH EDGE



- 2901 BASED

Figure 48. Median Filter Software

Object Features and Classifier--Object features are computed by CPU 1 on the median filtered bins in a hierarchical fashion. That is, less expensive features are computed initially to do preliminary clutter screening, and more expensive features are computed on the unrejected objects in order to do additional clutter screening and object recognition. Table 9 describes some features that currently are candidates for preliminary clutter screening; Table 10 lists features that currently are being used to do finer target discrimination. The intensity moments are by far the most expensive and are computed only on those objects for which it is absolutely necessary to do so. Clutter rejection and target classification is performed by a tree-structured classifier consisting of thresholding on simple features and features combined in discriminant functions. The features themselves are discussed in a previous interim report.[4]

[4]Soland, pp. 81-100.

TABLE 9. PRELIMINARY FEATURES AND CLUTTER PRESCREENING

- Simple features are computed for use in early clutter discrimination

- Four features per object
  - Area
  - Total brights
  - Average width
  - Average intensity

- Involves: 3 additions (approx.) per interval
  2 multiplications, 4 comparisons per object

- Clutter objects thrown out

As was mentioned previously in the bin matching subsection, object features computed by CPU 2 on the bins in the interval memory will also be stored in these bins. CPU 2 is notified of the location, size, and classification of targets detected by CPU 1. This can be done by having CPU 1 store the latter information in a fixed set of locations in the interval memory, interrupt CPU 2, and then allow CPU 1 to access the information via direct-memory access.

TABLE 10. PRIMARY FEATURES AND CLASSIFIER

- Features include those from preliminary screening plus:

  Total edges

  Average background intensity

  Moments

  Length/average width

  Average target contrast

  Perimeter/$\sqrt{\text{Area}}$

  Object type (hot or cold)

- Moments are by far the most expensive

  $\mu 11$, $\mu 20$, $\mu 02$, $\mu 21$, $\mu 12$, $\mu 30$, $\mu 03$ centroids (worst case)

- Classifier does detailed clutter rejection and target recognition

  -- Hierarchical

  -- Discriminant computations use 10 features

<u>Computational Requirements</u>

Estimates of the worst case computational requirements for CPU 1 are summarized in Table 11. These estimates were made by first estimating the amount of time needed to perform arithmetic operations (e.g., adds, subtracts, etc.) with all operands present, and then doubling this time to account for memory accesses needed to fetch the operands. Because CPU 1 is a microprogrammable processor, all estimates of computational requirements are initially given in terms of number of microinstructions or microcycles which are then converted to nominal execution times.

TABLE 11. CPU 1 WORST CASE COMPUTATIONAL REQUIREMENTS

● Bin matching + median filter		
50 objects x 32 intervals/object x 30 μ|/interval	= 48K μ |	(9.6 ms)
● Features, excluding moments		
50 objects (each 32 x 32) x 760 μ |/object	= 38K μ |	(7.6 ms)
● Moment features		
10 objects (each 32 x 32) x 5.7K μ|/object	= 57K μ |	(11.4 ms)
● Frame classifier		
10 objects x 200 μ |/object	= 2K μ |	(0.4 ms)
	145K μ |	(29.0 ms)
+ 100% overhead	145K μ |	(29.0 ms)
	290K μ |	(58.0 ms)

⇨ ~ 60% LOADING USING 200 ns μ CYCLE

It was assumed that in a worst case situation, CPU 1 would initially have to process 50 objects, each with a maximum size of 32 by 32 pixels. Bin matching and median filtering would be done on these 50 objects, along with the computation of nonmoment features. Clutter screening by the classifier would reduce this number to ten 32 by 32 objects on which moment features would be computed for further clutter screening and object recognition.

Some justification for the estimates in Table 11 is offered below.

Bin Matching and Median Filtering--It is estimated that approximately 30 microinstructions will be required for each interval to do the seven or so comparisons required to match a bin, and the four comparisons required to do the median filter and the bookkeeping functions required in bin matching (e.g., midpoint updating, etc.).

Features (excluding moments)--The mix of arithmetic operations on a 32 by 32 pixel object is as follows for various features:

Area (Σ width)	32 adds
Σ Edge	32 adds
Σ Bright	32 adds
Average background	32 adds, 1 division
Average intensity	32 adds, 1 division
Length/average width	1 division
Average width	1 division
Average target contrast	1 subtraction
Perimeter^2/area	64 adds, 64 subtractions, 64 absolute values, 1 division

Assuming it takes 1 microinstruction to do additions and subtractions, 3 microinstructions to do absolute values, and about 64 microinstructions to do a 16-bit division, the latter operations amount to about 760 microinstructions per object.

Moments--The moment feature calculations use additions, subtractions, multiplications, divisions, and table look-ups. In making estimates, it was assumed that 16-bit by 16-bit multiplications would be done by using a multiplier/accumulator chip external to the CPU 1 arithmetic logic unit, taking three or four microinstructions per multiplication. Without going into a detailed breakdown of the operations themselves, the estimate of the number of microinstructions required to do the arithmetic operations for the moments on a 32 by 32 pixel object breaks down as follows:

Centroids	262	microinstructions
$\mu02$, $\mu20$	352	"
$\mu11$	2275	"
$\mu30$, $\mu03$	320	"
$\mu21$, $\mu12$	2464	"
	5673	microinstructions

or approximately 5.7K microinstructions per object.

Frame Classifier--Assuming seven 10-feature discriminant functions, each involving 10 multiply/accumulates, the estimate is 200 microinstructions per object.

Summary--The totals are given in Table 11, including overhead. Assuming 200 ns per microinstruction, the total time to do the calculations is estimated at 58 ms, which is approximately 60 percent of the available 100 ms (0.1 sec). Note that 50 non-overlapping 32 by 32 pixel objects cannot physically coexist in a frame using the PATS 512 by 256 sampling grid. This fact tends to lend some credibility to the assertion that these are worst case estimates.

CPU 2 FUNCTIONS

The processing requirements of CPU 2 are modest compared to those of CPU 1. Software functions for CPU 2 are shown in Table 12 and in the block diagram in Figure 44.

Interframe analysis and symbol generation are the only functions which CPU 2 must perform in a field operational system. The other functions, i.e., diagnostics, feature dumps, and test data generation, are necessary for debugging the hardware and software in the laboratory or bench version.

TABLE 12. CPU 2 SOFTWARE FUNCTIONS

- Memory diagnostics
- Feature dumps for off-line training
  - Interval features
  - Object features
- Test data generation
- Interframe analysis
- Symbol generation

Interframe Analysis and Symbol Generation

The interframe analysis accumulates the classification results of at most 10 objects from each frame processed by CPU 1, tracks these objects on a frame to frame basis, and outputs a cumulative classification for each of them. Object tracking is done only on a symbolic basis; that is, instead of intensity information, only object class, size, and position are used. Interframe analysis has been described in a previous report.[1]

Symbol generation refers to the generation of a graphics overlay on the operator's video monitor (see Table 13). This overlay identifies and highlights targets that were found by CPU 1 and classified by interframe analysis. A dot graphics memory covering an entire frame is maintained by CPU 2 and its contents are continually read out and mixed with the analog video. CPU 2 maintains the graphics memory essentially by erasing and rewriting it every sixth frame. Target highlighting techniques may include drawing a box around each target and generating some sort of identifying symbol; there are several options for highlighting since shapes and symbols will be defined by software and not hardware.

TABLE 13. SYMBOL GENERATION SOFTWARE

- Generates target cues on video display
- X, Y, target type, target extents passed from CPU 1 for each object
- Target enclosures generated by setting bits in dot graphics memory

Diagnostics, Feature Dumps, and Test Data Generation

These functions involve writing software to interface with the memories in CPU 1, as shown in Figure 44. Integrity of these memories can be checked by CPU 2; features generated by CPU 1 which are stored in the interval memory can be dumped to disk or tape for checking CPU 1 software integrity and off-line derivation of classifier coefficients. CPU 2 can also generate test data for CPU 1 and store them in the interval and intensity memories.

A detailed test of the PATS software can be achieved as follows. A frame is processed using a version of the PATS simulation designed to match the implementation software. All data generated by the simulation are saved, including interval data, object features, and classification results. CPU 2 is used to load the interval memory with the interval data, and the intensity memory with the digitized frame. Next, CPU 1 is run on the data stored in the interval and intensity memories (the autothreshold hardware is not used); CPU 2 is used to dump the results generated by CPU 1 for comparison with the simulation results. In another mode, CPU 2 functions can be checked by using diagnostics internal to CPU 2 and by allowing CPU 2 to generate its graphics overlay on a video disk playback of the frame processed by the simulation.

CPU 1 is a special purpose high-speed microprogrammed processor designed to meet a required processing rate of 10-video fields per second. CPU 2, on the other hand, is a more general purpose, off-the-shelf, "macroprogrammed" processor with far less computational loading than CPU 1. A special requirement of CPU 2 is that it interface with peripheral devices that provide off-line storage (e.g., floppy disk), character data input (e.g., CRT), and hardcopy output (e.g., line printer). This is because in addition to providing operational functions (interframe analysis, symbol generation), CPU 2 also supports diagnostics which provide a window into the operational software and hardware.

CPU 1 Software Development

CPU 1 will be programmed at the microinstruction level, and as such requires special development tools. A microassembler and text editor will be used to allow microcoding at the mnemonic rather than the bit level. Frequent changes will be made to the microcode during the development period, so during that time, the microcode will reside in a fast RAM or writable control store (WCS); PROM will be used in the operational system. The microcode development facility will provide an interface with the WCS and will allow single-step execution and address tracing of microcode during execution.

CPU 2 Software Development

CPU 2 will be programmed like a typical minicomputer, i.e., using an assembler and, if available, a higher order language. The interframe analysis simulation is coded in FORTRAN, so using FORTRAN will speed development of interframe analysis implementation software. Because CPU 2 will be interfaced with sophisticated peripherals (e.g., floppy disk), it would be very helpful if I/O drivers for these peripherals were provided by the manufacturer in a real-time operating system. The software development tools for CPU 2, unlike CPU 1, can actually be resident on CPU 2, thereby removing the problem of loading the CPU 2 memory with program code.

SECTION V

PLANS FOR THE NEXT REPORTING PERIOD

During the next reporting period, further refinements of the target screening algorithms are planned. These refinements include improvements in 1) segmentation, 2) clutter rejection, and 3) recognition algorithms. Improved segmentation is desirable to provide a better outline of the target for moment calculations. In particular, a merging of hot and cold bins corresponding to different parts of the same target is necessary. We intend to refine the feature selection for the clutter rejection classifier to provide a simpler and more robust clutter filter. We also intend to simplify the recognition classifier by further feature selection experiments. Also, the k-nearest neighbor classifier for recognition will be investigated. The use of k-nearest neighbor will simplify retraining of the classifier with new training samples.

Functional design specifications will be written for each of the hardware and software system elements. We also will begin detailed design of the timing and synchronization circuits, CPU 1 and its associated memory 1 and memory 2.

Checkout of the image enhancement breadboards with the MODFLIR will also be completed during the next reporting period.

REFERENCES

[1] D. E. Soland, et al., "Prototype Automatic Target Screener," Quarterly Progress Report, Contract No. DAAK70-77-C-0248, ADA 050684, January 15, 1978.

[2] Soland, p. 17.

[3] Texas Instruments Inc., "Operation and Maintenance Instructions with Parts List for Multipurpose Observation Device," September 1976.

[4] Soland, pp. 81-100.